PRIORITIZING SATELLITE PAYLOAD SELECTION VIA OPTIMIZATION

THESIS

Benjamin S. Kallemyn, Captain, USAF

AFIT/GOR/ENS/07-14

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense or the United States Government.

AFIT/GOR/ENS/07-14

PRIORITIZING SATELLITE PAYLOAD SELECTION VIA OPTIMIZATION

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Benjamin S. Kallemyn, B.S.

Captain, USAF

March 2007

AFIT/GOR/ENS/07-14

PRIORITIZING SATELLITE PAYLOAD SELECTION VIA OPTIMIZATION

Benjamin S. Kallemyn, B.S.

Captain, USAF

Approved:

Dr. Jeffrey P. Kharoufeh Thesis Advisor	Date
Major Gary W. Kinney, Ph.D. Committee Member	Date
Dr. David P. Morton Committee Member	Date

AFIT/GOR/ENS/07-14

Abstract

This thesis develops optimization models for prioritizing payloads for inclusion on satellite buses with volume, power, weight and budget constraints. The first model considers a single satellite launch for which the budget is uncertain and constellation requirements are not considered. Subsequently, we include constellation requirements and provide a more enhanced model. Both single-launch models provide a prioritized list of payloads to include on the launch before the budget is realized. The single-launch models are subsequently extended to a sequence of multiple launches in two cases, both of which incorporate an explicit dependence on the constellation composition at each launch epoch. The first case ignores future launches and solves a series of independent single-launch problems. The second case considers all launches simultaneously. The optimization models for single- and multiple-launch cases are evaluated through a computational study. It was found that, when the budget distribution is skewed, the prioritization model outperforms a greedy payload selection heuristic in the single-launch model. For the multiple-launch models, it was found that the consideration of future launches can significantly improve the objective function values.

Acknowledgements

I must acknowledge a handful of people whose help and support have been invaluable during this thesis process. First, thank you to my advisor, Dr. Jeffrey Kharoufeh, whose guidance and motivation were instrumental in the completion of this work. Second, thanks to Dr. David Morton for his many insightful comments and suggestions on this research. Likewise, many comments and discussions with Major Gary Kinney were indispensable. Next, thank you to Mr. Justin Comstock for the providing guidance on the realistic aspects of the problem. To my family, thank you for always being there and for being understanding throughout our time at AFIT. Finally, I must credit God with giving me the strength, wisdom, patience and perseverance to complete this thesis.

Benjamin S. Kallemyn

Table of Contents

Page

List of Figures

List of Tables

PRIORITIZING SATELLITE PAYLOAD SELECTION
VIA OPTIMIZATION

1. Introduction

1.1 Background

The development, deployment, and maintenance of satellite systems is a significantly costly endeavor. In fiscal year 2006, the Department of Defense (DoD) requested a budget of "more than $23 billion to develop, acquire, and operate satellites"[12]. While it is not surprising that the military spends large amounts of money on its programs, there is also significant spending on satellites in the private telecommunications sector. According to its 2006 annual report, EchoStar®, a commercial satellite communications corporation, spent more than $112 million on satellite and transmission expenses in 2005 (see [8]). Clearly, satellites are critical assets needed to accomplish certain missions in both the public and private sectors. Selecting the right mix of capabilities to include on a satellite, subject to cost and mission effectiveness constraints, is paramount. Because smaller, more powerful satellites require cutting-edge technology, acquiring and launching satellite systems is quite expensive. Moreover, satellite systems are launched relatively infrequently, and the systems are custom built to specification. The proper implementation of methodologies to effectively assign payloads to satellites can drastically reduce the overall development, deployment, and maintenance costs.

Satellites play an assortment of roles both in the military and in the civilian sphere of operations. The military maintains navigation, imaging, reconnaissance, weather and communication satellites, among others. Meanwhile, telecommunications companies offer satellite television, radio, internet and telephone services.

Weather forecasting and climate and environmental monitoring (done by many research agencies) are areas in which satellites play an integral role as well. Because satellites are used in such a wide array of applications, it will be beneficial to provide a general framework within which decision makers can prioritize capabilities that should be incorporated on satellites.

Satellites and satellite constellations are designed to perform specific missions. The specific items included on a satellite, usually referred to as *payloads*, perform specific functions (e.g, navigation, data communications, surveillance, etc.). The satellite bus houses all of the payloads, the power supply, flight computer and other necessary equipment not related to the payload. The physical space on the bus is limited as is the available power and weight. The power supply in the bus provides electricity to the payloads and is limited by the type of power supply and the amount of power required by the non-mission functions of the satellite. Deciding which payloads to include on new satellite launches is a nontrivial problem. Moreover, when the number of payload alternatives is large, the budget resources are uncertain, and sequential satellite launches exhibit functional or economic dependencies, the problem becomes even more difficult.

A *constellation* is a group of satellites working in concert to "cover" a certain geographical region and to accomplish specific mission requirements. For instance, for the Global Positioning System (GPS) to generate a location, a constellation of satellites in an area is needed to triangulate that location. Constellations are constructed and maintained by establishing a schedule of staggered satellite launches that carry payloads into the constellation. Certain payloads enhance the effectiveness of the constellation to perform its mission; however, at some point, the launch of certain additional payloads yields diminishing returns. Deciding which payloads to include on specific satellite launches based on the characteristics of the payload, capacities of the satellite, and composition of the constellation, is of utmost importance.

Generally, the payload selection problem is solved either by a committee in an ad-hoc manner or it is solved analytically. A group of subject matter experts can decide which payloads to include on a satellite by coming to an agreement on the value each payload brings to the constellation. Payloads are usually selected for inclusion on the satellite in order of greatest importance as deemed by the committee. Another approach is to select payloads with the intent of optimizing some measure of importance. Common approaches include simulation modelling and optimization. The use of a mathematical approach provides decision makers with an unbiased viewpoint as to which payloads best meet the specific constellation objectives.

In this thesis, we provide optimization models to prioritize the candidate payloads for inclusion on a satellite bus being launched into a pre-existing satellite constellation. We will develop realistic models that allow for uncertain resource levels to improve the existing optimization methods currently used. We also model the payload prioritization problem from the viewpoint of maintaining mission requirement levels as opposed to the viewpoint of maintaining payload operability. These models will allow decision makers to differentiate between the importance of satellite payloads via the use of priority levels.

1.2 Problem Definition and Methodology

Consider a constellation into which a sequence of satellites are launched at fixed, equal time intervals. Payloads which best satisfy the mission requirements of the constellation are selected from a list of available payloads for inclusion on the bus of each satellite to be launched. For each satellite launch, there exist multiple, potential life-cycle budget scenarios and the payloads must be selected for inclusion on the satellite bus prior to the budget realization. The main purpose of this thesis is to provide optimization models which may be used to prioritize the multiple alternative payloads for each satellite bus in a sequence of launches such that the reward to the

constellation is maximized. Using the solutions of the models, a decision maker can select payloads for inclusion on satellite busses at each of the various launch epochs.

The payload prioritization problem is first modelled, in its simplest form, as a knapsack problem to represent a single satellite with a certain budget. The knapsack problem seeks to fill a backpack with candidate items so as to minimize the total cost of the items while at the same time maximizing the space occupied by the items placed in the pack. The satellite bus is considered as the knapsack and is constrained by the available budget and the weight, power, and volume capacities of the bus. The available payloads are selected for inclusion based on their reward value. This simplistic single-launch model is extended to include uncertain budget levels, to assign priority levels, and to satisfy mission requirements. The resulting linear, integer mathematical programming model realistically captures the essence of the payload prioritization problem for a single satellite launch. The single-launch model can be solved using the branch-and-cut algorithm used in the CPlex solver by ILOG®. The performance of this model will be tested by comparing the results of randomly-generated problem instances with a simplistic payload selection strategy.

The single-launch prioritization model is subsequently extended to account for a sequence of launches that carry payloads to the constellation in order to satisfy the constellation's mission requirements. Rewards are accrued for payloads included on each satellite bus based on the satisfaction of mission objectives by the constellation. We consider two models, both of which account for the current composition of the constellation. The first considers each satellite launch sequentially and does not account for the reward values on subsequent launches. The second model simultaneously selects payloads for all launches in the time horizon to maximize the reward accrued for the entire launch horizon. Each of the multiple-launch models can be solved using the branch-and-cut algorithm of CPlex solver by ILOG®. We will evaluate the performance of the two model types by comparing the objective

function values of the model which considers the future with the model that does not.

The main objective of this thesis is to provide an optimization tool with which analysts can prioritize items to include in their respective "knapsacks" in the face of uncertain budgets. The payload prioritization models developed herein can be solved analytically using an off-the-shelf LP solver. This optimization technique for prioritizing items finds wide applicability in both the public and private sectors, including, but not limited to, aircraft loading, resource allocation, delivery route selection and facility location problems.

1.3 Thesis Outline

The remainder of the thesis is organized as follows. In Chapter 2, knapsack problems, facility location problems and stochastic programming are reviewed along with the current literature related to payload selection methodologies and prioritization models. Chapter 3 presents the assumptions and mathematical models for the single- and the multiple-launch payload prioritization models. Chapter 4 describes the computational experiments conducted using random problem instances of the payload prioritization problem and summarizes their results. Finally, Chapter 5 provides the conclusions of this thesis, some recommendations for using the models, and suggestions for future extensions of the work.

2. Relevant Literature

This chapter discusses the literature pertinent to the selection of payloads to include on a satellite bus. The first section reviews models that are used in this thesis: the knapsack problem, the facility location problem and stochastic programming models. The next section examines some existing payload selection methodologies found in the literature. Finally, we conclude this chapter by discussing prioritization models.

2.1 General Modelling Review

We now review some well-known optimization models pertaining to the research approach of this thesis. First, we examine the knapsack problem (KP) and variations thereof. Next, we review the facility location problem (FLP) and its variants. Both of these problems are known to be NP-hard [11]. A brief description and the general mathematical programming model for each problem are provided. Finally, we give a brief overview of stochastic programming and its application to these models.

The knapsack problem (KP) [25] can be described as follows. Suppose a hiker has a knapsack that holds a maximum of W pounds. Let I denote a finite set of n items that the hiker can choose to carry in his pack. Each item, $i \in I$, has a weight (w_i) and a value (v_i) to the hiker. The hiker selects items to place in the knapsack so as to maximize the total value of the items in the pack. For each item i, the decision variable x_i assumes a value of 1 if item i is included in the knapsack and 0

otherwise. The mathematical programming formulation follows:

$$\max_{x} \quad \sum_{i \in I} v_i x_i \tag{2.1a}$$

$$\text{s.t.} \quad \sum_{i \in I} w_i x_i \leq W, \tag{2.1b}$$

$$x_i \in \{0, 1\}, \qquad i \in I. \tag{2.1c}$$

The objective (2.1a) is to maximize the total value to the hiker. Constraint (2.1b) limits the total weight of the knapsack to its capacity, and (2.1c) is a binary restriction on x_i; either item i will be in the knapsack or it will not. Partial items are not permissible.

A well-known variant of the KP is the multidimensional knapsack problem (MDKP), in which there is more than one constraint (resource) affecting the selection of items for inclusion in the knapsack, i.e., we have m resources as opposed to just the one for weight. Let J denote the set of m resources. For $j \in J$, let b_j denote the capacity of resource j and a_{ji} denote the consumption of resource j by item i. The decision variable x_i again assumes a value of 1 if item i is included in the knapsack and 0 otherwise. The formulation is as follows:

$$\max_{x} \quad \sum_{i \in I} v_i x_i \tag{2.2a}$$

$$\text{s.t.} \quad \sum_{i \in I} a_{ji} x_i \leq b_j, \qquad j \in J \tag{2.2b}$$

$$x_i \in \{0, 1\}, \qquad i \in I. \tag{2.2c}$$

The objective (2.2a) maximizes the total value of the knapsack. Constraints (2.2b) allow only items that fit within the capacity of the knapsack to be selected for inclusion. As in the KP formulation, the binary constraint does not allow partial items to be selected for inclusion in the knapsack. Both the knapsack problem and

the multidimensional knapsack problem can be solved to optimality using an off-the-shelf solver such as the branch-and-cut algorithm in CPlex solver by ILOG®.

We now review the well-known facility location problem (FLP) [25]. Suppose we must satisfy customer demand by choosing facility locations, e.g., we are building a chain of stores (locations) to serve some geographical area. Let $l \in L$ denote the set of facility locations and $k \in K$ represent the set of customer demands. Denote by v_{lk} the value obtained by serving customer k with location l. We select facility locations so as to maximize the total value of serving demands with the facilities used. We have a resource budget, b, and facility l uses c_l units of the resource. For each location l and customer demand k, the decision variable x_{lk} assumes a value of 1 if facility location l serves customer demand k and 0 otherwise, and the decision variable y_l assumes a value of 1 if facility location l is used and 0 otherwise. The mathematical programming formulation is as follows:

$$\max_{x,y} \quad \sum_{l\in L}\sum_{k\in K} v_{lk}x_{lk} \tag{2.3a}$$

$$\text{s.t.} \quad \sum_{l\in L} c_l y_l \le b \tag{2.3b}$$

$$\sum_{l\in L} x_{lk} = 1, \qquad k \in K \tag{2.3c}$$

$$x_{lk} \le y_l, \qquad l \in L, k \in K \tag{2.3d}$$

$$x_{lk} \in \{0,1\}, \qquad l \in L, k \in K \tag{2.3e}$$

$$y_l \in \{0,1\}, \qquad l \in L. \tag{2.3f}$$

The objective (2.3a) maximizes the total value of serving customers with the facilities. Constraint (2.3b) limits the locations selected to the resource capacity, (2.3c) ensures each customer demand is filled exactly once, (2.3d) ensures that only selected facilities will be used to fulfill demand, and (2.3f) and (2.3e) are binary restrictions on x_l and y_{lk}, respectively. This model allows facilities to satisfy multiple customers.

It is noteworthy that the FLP has an embedded KP; i.e., Constraints (2.3b) and (2.3f) are of the same form as Constraints (2.1b) and (2.1c).

We use stochastic programming to introduce uncertainty into mathematical programming models [2]. This allows us to model real-world problems more accurately since they are seldom entirely deterministic. We must make a set of decisions, known as first-stage decisions, prior to the realization of the random information. Any decisions that adjust the a priori decisions, after a realization of the random parameters, are known as second-stage, or recourse decisions.

Using the context of a knapsack model as our base case, we illustrate how stochastic programming adds uncertainty to a model. Suppose a hiker is borrowing a backpack and does not know which of several backpack styles he will receive upon arrival to a rental location. Each backpack type has a different weight capacity. Let Ω represent the finite set of the possible (backpack capacity) scenarios. Assume each scenario $\omega \in \Omega$, has an associated probability mass, q^{ω}. Let I denote the finite set of n items being considered for inclusion. The hiker must select from among the n initially available items those to include in his backpack before he rents it and subsequently realizes its capacity; this is the first-stage decision. Each item, $i \in I$, has a weight (w_i) and a value (v_i) to the hiker. We assume in this illustration that the inclusion of the first stage items will not exceed the weight of the smallest backpack, denoted b^1.

Let J denote the finite set of m additional items available in the rental shop that can consume the space in the pack not filled by the items decided upon in the first-stage and which increase the total value of items in the backpack. Each item $j \in J$ has a weight of w_j and a value of v_j to the hiker. The decision of which additional items from the shop to include is the recourse decision. For each item i, the first-stage decision variable x_i takes a value of 1 if item i is included in the backpack and 0 otherwise. The recourse variable y_j^{ω} assumes a value of 1 if item j is selected for inclusion in the pack under scenario ω and 0 otherwise. The

mathematical programming formulation follows:

$$\max_{x,y} \quad \sum_{i\in I} v_i x_i + \sum_{j\in J}\sum_{\omega\in\Omega} q^\omega v_j y_j^\omega \tag{2.4a}$$

$$\text{s.t.} \quad \sum_{i\in I} w_i x_i + \sum_{j\in J} w_j y_j^\omega \le b^\omega, \qquad \omega \in \Omega \tag{2.4b}$$

$$\sum_{i\in I} w_i x_i \le b^1, \tag{2.4c}$$

$$x_i \in \{0,1\}, \qquad i \in I, \tag{2.4d}$$

$$y_j^\omega \in \{0,1\}, \qquad j \in J, \omega \in \Omega. \tag{2.4e}$$

The objective function (2.4a) maximizes the sum of the expected reward from the first-stage decision variables and the sum of the reward from the recourse variables. Constraints (2.4b) limit the total weight of each backpack type to its capacity. Constraint (2.4c) ensures that the items selected prior to the realization of the backpack size do not exceed the weight of the smallest backpack. Constraints (2.4d) and (2.4e) are binary restrictions on the decision variables; i.e., partial items are not permissible in the backpack. It is possible to formulate the multidimensional knapsack problem and the facility location problem as stochastic programs by combining Formulation (2.4) with Formulations (2.2) and (2.3), respectively.

The solution to the stochastic programming formulation, i.e. Model (2.4), contains two lists. The first corresponds to the items that you take to the rental location that will fit into all available backpacks. These items are determined based on the decision variable x_i. Upon realizing the backpack size rented, the decision variable y_j^ω indicates which items should be purchased at the shop in order to maximize the remaining space in the bag, given the backpack rented.

As we will show in the next section, the knapsack problem and the multidimensional knapsack problem lend themselves well to the payload selection problem. The candidate payloads can be viewed as items being considered for inclusion in the

knapsack, or satellite bus. The payload selection problem can also be thought of as a facility location problem. We let the satellite mission objectives correspond to the customer demands and the available payloads that will be launched to satisfy those objectives correspond to the facility locations. We will use this framing of the facility location problem in Chapter 3. Likewise, we will introduce uncertainty into the payload selection model in the next chapter using stochastic programming to make the model more tenable.

2.2 Satellite Payload Selection

Central to this thesis is an understanding of the current methodologies used to select payloads for inclusion on satellite buses being launched into constellations. The literature on when to launch satellite payloads based on reliability theory is abundant; however, we are more concerned with the methodologies for selecting payloads on satellites being launched at predetermined times. We will begin by reviewing the design of satellite constellations followed by a review of methods used to select payloads for individual satellites.

Diekelman [7] described a constellation design methodology based on business objectives rather than technical aspects of the design. The potential for revenue from the timeframe, services and market drive the design process. Based on these drivers, the technical requirements (bandwidth, coverage, capacity, performance, etc.) are derived. Design parameters, e.g., solar radiation, space debris, orbital geometry, orbital path, etc., influence which candidate constellations are selected based on an analysis of alternatives. After the decision of which constellation to utilize is made, the design is fine-tuned to include secondary effects, i.e., orbit parameters associated with the gravitational pull of the moon. The six requirements which dominate the constellation selection process are (in order of importance): service area coverage, spectrum sharing (bandwidth availability), capacity augmentation (handling high usage for short time periods), satellite failure mitigation, service

link maintenance (hand-offs of service between satellites) and altitude considerations (altitude, elevation angle, radiation, etc.). Each of the above requirements is assessed based on the potential revenue generated.

Most design methodologies focus only on coverage, i.e., how to limit the number of satellites in the constellation and still achieve coverage of the desired area. Lansard and Palmade [16] presented a methodology for designing constellations via multi-criteria decision making. Using an optimization approach that minimizes the number of satellites subject only to geographical coverage is not sufficient to optimize a constellation. This approach leaves the constellation susceptible to failure since the minimization of the number of satellites eliminates redundancy. The multi-criteria decision making model handles three objectives: coverage (number of satellites), availability (redundancy, capacity, production rates, etc.) and life-cycle costs. The solutions for multi-objective problems vary greatly depending on the relative importance of each objective. A cost-effectiveness approach provides optimal constellation design parameters in terms of constellation coverage and availability.

Wertz and Larson [24] discussed the elements of constellation design, but they also suggested a process for selecting payloads on satellites. This process of defining (selecting) a payload flows in the following manner. First, payload objectives are based on the mission objectives and requirements. Then the performance thresholds of the objectives are established, including how the end-user interfaces with the payload. Based on the objectives, candidate payloads are identified and analyzed by estimating candidate payload characteristics via analogy with existing systems, scaling from existing systems and budgeting by components (the overall system is estimated by the sum of the parts). Payloads are compared over key performance measures, e.g., life-cycle cost, quality of payload, performance, etc. The selection of satellite payloads based on analysis is difficult given the inability to quantify the benefits of some design characteristics; therefore human insight serves as the final judge as to which payloads are selected. While this general process guides

the selection of payloads, it offers no formal optimization techniques for payload selection.

Jacobs, *et al.* [14] described a methodology using Monte-Carlo simulation that indicates which payloads to launch and when to launch them. This model, Operational Constellation Availability and Reliability Simulation (OSCARS), is used by the U.S. Space Command Launch Services Office. OSCARS requires several inputs: the lifetime distributions for each payload, the state of existing payloads at start of the simulation, the availability of each payload and of launch vehicles over the time horizon. In the simulation, payloads are launched into the constellation, subject to availability and capacity restrictions, when an existing satellite in the constellation fails. The output from OSCARS is a series of graphs and tables. The most important output is the $x\%$ Launch Need Date which means that in $x\%$ of the replications, the launch occurred on or before this date. Based on this figure, the decision maker can schedule launches for each payload over the entire time horizon.

Bell [1] developed a model that selects 24 of 29 (in 1999) Global Positioning System (GPS) satellites to detect nuclear detonations in the Earth's atmosphere via its secondary payload, a Nuclear Detonation Sensor (NDS). The model is a specialized knapsack problem in which the constraint is the limit of 24 satellites. The reward (calculated using a classified simulation model) is a composite of the total coverage of the Earth's surface, the satellite's orbital location and the type of the nuclear sensor. A heuristic search algorithm provides a list of the 24 "best" satellites to monitor for nuclear detonation monitoring.

Brown, *et al.* [3] presented a capital-planning model which optimally selects the best candidate space systems to meet the requirements of U.S. Space Command over a 24-year span. This tool, called Space and Missile Optimization Analysis (SAMOA), is a collection of several analyses used to select which space systems to fund from hundreds of candidate systems. Space Command Optimizer of Utility Toolkit (SCOUT) is the linear-integer optimization model that selects the portfolio of

payloads, launches and funding period subject to several budget, system operational and time-line constraints. The objective of SCOUT is to maximize the number of requirements met by minimizing penalties for violating certain constraints (i.e., elastic constraints). These models were used to shape Space Command's strategic master plan in 1997 and 1999. The SCOUT model may be solved using commercial optimization software on a personal computer in less than one hour.

Flory [10] developed a payload selection method using a multidimensional knapsack (MDKP) model. The model uses the relative utility of each payload as the reward for including the payload on the satellite bus. The relative utility is calculated using three inputs: relative importance of the payload, the mean mission duration (MMD) of the payload and the number of functional payloads of that type in the constellation. The constraints for the MDKP correspond to the engineering specifications of weight, power and volume as well as a finite (known) budget. The integer programming model was then extended to account for multiple launches and solved to optimality using a commercial solver (XPRESS®). The model is also formulated as a dynamic program and solved using an enumeration algorithm in MATLAB®. These exact solutions were then compared with several heuristics (simulated annealing, greedy and two norm-based methods) which were also solved using MATLAB®.

Farias and Van Roy [9] considered a dynamic resource allocation problem to maximize the average utility over T time periods. Two linear integer models and their linear programming relaxations were formulated. An optimal vertex approximation algorithm, a randomized rounding approximation algorithm and a task-assignment heuristic were compared for various time and resource levels. For large problem instances the heuristic method fails to produce a feasible solution, while the two algorithms performed comparably with respect to computation time.

The methodologies used in constellation design give insight as to how we can determine potential constellation requirements which may be satisfied by the payloads

being launched. We use this concept to extend the state-of-the-art by embedding a knapsack problem, which has been shown to be useful in payload selection, into a facility location problem to satisfy the constellation requirements. To our knowledge, this approach is unique for the payload selection problem.

2.3 Prioritization Models

Prioritization is the process of generating an ordering, a priority list, of items that will be sequentially selected until a given resource is consumed. If the budget level for the resource allows six items to be selected from the list, we take the first six items on the priority list. If the budget allows only two, the first two items are chosen, etc. The item of interest may be the items in a knapsack problem or the cities to visit in a prize-collecting travelling salesperson problem.

Jung [15] developed a procedure that partitions a budget range for a knapsack problem and identifies the associated optimal set of items to include under each partition. The procedure consists of two steps that are repeated until the entire budget is partitioned. The two integer programming formulations (one for each step) follow:

$$\begin{aligned} \max_{x} \quad & Z_0 = \sum_{i \in I} v_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} c_i x_i \leq b \\ & x_i \in \{0,1\}, \qquad\qquad i \in I. \end{aligned} \tag{2.5}$$

$$\begin{aligned} \min_{x} \quad & Y_0 = \sum_{i \in I} c_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} v_j x_i = Z_0^* \\ & x_i \in \{0,1\}, \qquad\qquad i \in I. \end{aligned} \tag{2.6}$$

The first step is to solve (2.5) with $b = b_{max}$, the maximum budget. The resulting optimal solution and value are x^* and Z_0^*, respectively. The second step is to solve (2.6) using the optimal value from the first step. The first partition of the budget is $Y_0^* \leq b \leq b_{max}$ with optimal solution x^*. Finally, the first step is repeated with $b = Y_0^*$ and the inequalities changed to strict inequalities. The process continues until the minimum budget level is reached. This methodology is effective when an uncertain budget will be realized early in the decision making process and the selected items do not have to be consistent between various budget levels.

Morton *et al.* [19] developed stochastic network interdiction models that may be used to determine the best sites at which to install nuclear detection sensors. Two models were introduced, the stochastic network interdiction problem (SNIP) and the perceived stochastic network interdiction problem (PSNIP). The objective of both models is to minimize the probability that a smuggler will successfully traverse the network without being detected. The difference in the models is that, in SNIP, both the smuggler and the interdictor "agree" on the probabilities of detection, while in PSNIP, the smuggler and the interdictor have different perceptions about the probabilities of detection, e.g., the smuggler may not be aware of all the sensor locations. Both models were implemented on bipartite networks where sensors are only located at border crossings of a single country. In the case of SNIP, decomposition, duality and reformulation were used to transform the problem into a tractable optimization model. Step inequalities were introduced to tighten the relaxation and reduce the computational time.

Pan and Morton [20] solved the network interdiction model, SNIP, using primarily the L-shaped decomposition method [23]. A heuristic was used at certain iterations of the decomposition to reduce the computational effort. Also, valid inequalities were developed to tighten the relaxed problem. Using the solution, decision makers may determine where to install nuclear material detection devices. While the solution to the model is not a prioritized list, the decision maker has enough information to generate one in a post-processing step. However, there is no method for ensuring the list of sensors is consistent in the presence of an uncertain budget.

Golden *et al.* [13] examined the orienteering problem, and Tang and Miller-Hooks [22] and Laporte and Martello [17] discussed a similar problem, the selective travelling salesperson problem. Consider a set of nodes that can be visited by the traveller only once. A reward is accrued for visiting each node. The number of nodes, or cities, a traveller can visit is limited by some predetermined distance or time, e.g., a traveller is limited to travelling 45 miles, but the shortest distance required to visit every city on his route is 75 miles. The orienteering problem determines the tour that maximizes reward subject to the limiting distance. This tour can be considered a priority list. The solution technique used in [13] was a three-step center-of-gravity heuristic. The steps are route construction, route improvement and a center-of-gravity step. An exact branch-and-cut algorithm and a construct-and-adjust heuristic were the solution techniques used in [22]. Laporte and Martello [17] formulated the problem as a linear integer program, developed upper and lower bounds for an exact algorithm and implemented the algorithm to analyze the results. Smith [21] outlined how the National Air and Space Agency (NASA) uses the orienteering problem to determine an ordered subset of the vast programs NASA plans which can be executed within a given time horizon and with limited resources. There appears to be no literature pertaining to the orienteering problem with an uncertain distance constraint, which would make a prioritized list of nodes to visit worthwhile.

Dean *et al.* [5, 6] discussed the value of adaptivity in stochastic packing and stochastic knapsack problems. In the stochastic knapsack problem, the objective is to maximize the total reward when the weight of each item is not known (except by its probability distribution) until it is selected for inclusion in the knapsack, at which time its value is realized. Items are included in the knapsack until the budget is exhausted; once an item's addition exceeds the available budget, no additional items can be added. The stochastic packing problem is similar except that the weights of the items are vectors, like in the MDKP. Feasible items are not allowed to exceed the capacity of any component. A non-adaptive model selects the items to include before the realization of the uncertain data. This can be viewed as a priority list which is used to select the items to include and the order in which they should be included. An adaptive model can be thought of as a priority list that is updated after each item was included and its value was realized. The adaptivity gap was evaluated, which was the ratio of the expected optimal values of the adaptive method to the non-adaptive method.

Mettu and Plaxton [18] examined the online median problem, which is a variant of the k-median problem. The k-median and online median problems are similar to the facility location problem. Consider a grid on which there is one customer at each intersection. Assume the customer's demand will be satisfied by the nearest store. The objective is to minimize the total distance travelled by customers to stores. The facility location problem determines the location of the stores assuming all stores will be built at once. The k-median problem determines the best location for k stores to be built all at once, where $0 < k < n$. The online median problem determines the locations of the stores and provides the order in which they should be built (assuming they are built one at a time) when the number of stores to be built is unknown. The objective of the online median problem is to minimize the maximum competitive ratio which is defined as the ratio of the cost of the first k ordered locations to the cost of the optimal k-median solution, where the maximum

is taken over all k. The problem was solved using an approximation algorithm, and the solution is, in essence, a priority list.

There are several areas in the state-of-the-art that use the idea of prioritization: the orienteering problem, the selective travelling salesperson problem and the stochastic packing problem. However, these problems do not include uncertain information, e.g., the cutoff for the orienteering problem is deterministic. The solution to the online median problem yields a priority list over some uncertain data, however, the model was developed using a worst-case analysis instead of assuming a probability distribution for the unknown parameter. The stochastic network interdiction problem is one example where a prioritization model is especially beneficial to the decision maker in the context of uncertain budget scenarios. It appears that very little work has been done in regard to formal optimization procedures for selecting satellite payloads. And so, in this thesis we will develop optimization models for selecting and prioritizing payloads to include on a satellite that will be launched into a pre-existing constellation.

3. Optimization Models for Payload Selection

This chapter presents mathematical programming formulations for optimally selecting payloads to include on satellites being launched into constellations. In particular, we prioritize satellite bus payloads to maximize the expected reward for including these payloads in the constellation. We first describe the general model assumptions and definitions for the payload prioritization problem. Next, we examine a simple single-launch model with a known deterministic budget. The deterministic single-launch model is then extended to a payload prioritization model with mission requirements and an uncertain budget. In the fourth section, we discuss multiple-launch models wherein the decision maker must select payloads over a finite time horizon. In this extended model, multiple, sequential launches occur over a specified time period and the rewards depend on the payloads already present in the constellation. As an illustrative example, we consider a telecommunications satellite throughout this chapter.

3.1 General Model Assumptions and Definitions

Consider a constellation into which a sequence of satellites are launched at fixed, equal time intervals. Let Δ denote the fixed inter-launch time. The number of functioning satellites and their respective payloads in the constellation are assumed to be known. We define L as the set of launches in the sequence and let the subscript l on any variable denote the dependence of the quantity on the given launch. The subscript is dropped in the single-launch case.

Payloads (or capabilities) are included on each satellite launch. Assume that only one payload of any given type may be included on a single launch. This restriction can be circumvented, if the need arises, by including the payload in the set of feasible payloads more than once. For instance, if it were possible for two high-gain antennas to be included on a satellite, the set of available payloads for the satellite

would have two instances of a high-gain antenna. The mean mission life of a satellite (and thus the payloads on the satellite) is assumed to be a fixed quantity which is a multiple of Δ. This mean mission life has the form $t\Delta$, where the scalar t is defined as the number of launches that occur during its life. The satellite's mean mission life accounts for the degradation of the components on the satellite over time. Rather than make the models more complex by including a degradation process, we assume that all satellites (and their payloads) are rendered useless t time units after they are launched. In the mathematical models, let K denote the finite set of n available payloads for any launch.

On each satellite, the satellite bus is constrained by its *engineering specifications*, i.e., the physical quantities that limit which payloads may be included in the satellite bus. Due to the nature of satellite design, we assume the engineering specifications are known with certainty and remain constant. Since there may be numerous engineering specifications which constrain the payload selection decision, the models in this thesis generalize all specifications with one constraint type. Of the many aspects of the satellite bus design which limit the inclusion of payloads, we will focus on three. First, the weight of the payloads which might be included on the satellite must be considered because the launch vehicles that lift the satellite into orbit have a finite weight capacity. The second engineering specification we include is power. The satellite bus allocates power to the included payloads from the satellite's power source which is recharged via solar panels. The power consumption of all payloads included on the satellite bus cannot exceed its total power output. Finally, we incorporate the engineering specification of volume. The actual space that is available for payloads in the satellite bus constrains which payloads can be included. Since the satellite will be fabricated subsequent to the payload selection decision, and to make the model more tractable, we account only for the volume required for each payload, ignoring any geometric considerations. The following sets and notation pertaining to the physical specifications will be used in the mathematical models to follow. Let

D denote the finite set of engineering specifications of interest for each launch in the sequence. Let u_{dl} denote the capacity for engineering specification d on launch l and let a_{kd} denote the consumption of engineering specification d by payload k.

Assume that there are m budget scenarios for each satellite launch. A budget scenario is a possible life-cycle budget level that can be expected for a given satellite launch. Let Ω denote the finite set of possible budget scenarios. For instance, Ω = {High, Med, Low} means that there are three possible budget scenarios: a high budget level, a medium budget level, and a low budget level. Each launch has an independent set of budget levels, where b_l^ω is the budget level for launch l under scenario ω. Assume budget scenarios b_l^ω have associated probability masses, q^ω, for $\omega \in \Omega$, and that without loss of generality, the budget scenarios are ordered such that $b_l^1 < b_l^2 < \cdots < b_l^m$. For instance, a high budget scenario could have a probability of 0.2, a medium, or average, budget level might occur with probability 0.6, and a low budget scenario with probability of 0.2. The number of budget scenarios and their respective probability masses are assumed to be equivalent over all launches. There is no monetary carryover from the budget of one launch to the next. All launches are assumed to be *temporally dependent*, i.e., each launch will see the same realization of ω. If the budget level for the first launch is high, every other launch will receive the high budget as well. In case the budget is known with certainty, the ω superscript is omitted. Let c_{kl} denote the life-cycle cost for including payload k on launch l. The total life-cycle costs of the payloads selected for inclusion on the satellite bus cannot exceed the budget level for any launch.

Assume that a separate priority list is generated for each launch since the budget levels for each launch are separate and there is no budgetary carryover between launches. Let $I = \{1, 2, \ldots, n\}$ denote the set of n priority levels for each launch. The cardinality of the set I is the same as that of the set K, i.e., the number of priority levels equals the number of available payloads for each satellite launch. Although there is a priority level for each payload, not every priority level is necessarily as-

signed. For every payload that is not included under the largest budget scenario for the launch, there is a priority level which is not assigned. The models will determine which payloads, and thus priority levels, are excluded.

3.2 Single-Launch Models with Certainty

We first consider the case of only a single satellite launch with a known budget. The budget and engineering specifications (weight, power, volume, etc.) constrain the number of distinct payloads that may be selected for inclusion on a satellite bus. We assume that the monetary budget for a single-satellite launch is known with certainty, i.e., the amount of money allocated for the satellite design is not susceptible to change. Furthermore, this framework is suitable if the budget is not known with certainty, but the payload selection decision can be postponed until the budget becomes known, e.g., in the case where the full project budget is awarded only after it is approved to be carried out. The single-launch problem with certain budget can be modelled as a multidimensional knapsack problem (MDKP). The objective is to select the set of payloads to include on a satellite bus in order to maximize the reward, given a certain monetary budget.

In this model, we assume the reward gained by including any one of the payloads on the bus is determined in advance and does not change. For $k \in K$, if payload k is included on the satellite, a reward r_k is accrued and the decision variable x_k assumes a value of 1 if payload k is included on the satellite and 0 otherwise. A summary of the problem data follows:

Sets:

$k \in K$	set of candidate payloads (capabilities)
$d \in D$	set of included engineering specifications (weight, power, volume, etc.)

Data:

r_k	reward received for including payload k
c_k	cost of payload k
b	maximum allowable budget for loading the bus
a_{kd}	consumption of resource d by payload k
u_d	capacity of resource d

Decision Variables:

x_k	1 if payload k included on the satellite bus; 0 otherwise

The mathematical programming formulation of this multidimensional knapsack problem (MDKP) follows:

$$\max_{x} \quad \sum_{k \in K} r_k x_k \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{k \in K} c_k x_k \leq b \tag{3.1b}$$

$$\sum_{k \in K} a_{kd} x_k \leq u_d, \qquad d \in D \tag{3.1c}$$

$$x_k \in \{0, 1\}, \qquad k \in K. \tag{3.1d}$$

Constraints (3.1b) and (3.1c) limit the inclusion of payloads to the available budget and engineering specifications of the satellite. The binary variables ensure that only a single unit of any payload type can be included. It should be noted that Constraint (3.1c) can include any set of generic specifications and allows this and subsequent adaptations of the model to be applied to other problem classes such as the facility location problem, the travelling salesperson problem or the orienteering problem.

We illustrate this simplistic single-launch model with a numerical example. We seek to launch a telecommunications satellite into an existing telecom constellation. The typical payloads included on this type of satellite are a receiving antenna, a

transmitting antenna and a transponder. Suppose there are two of each type of payload available for the next launch. Let $K = \{1, 2, 3, 4, 5, 6\}$ represent the set of available payloads as noted in Table 3.1 along with the reward, cost, weight, power and volume for each payload. Let $D = \{1, 2, 3\}$ denote the engineering specifications of interest (1 = weight, 2 = power consumption, 3 = volume). The available budget and the weight, power and volume capacities for the satellite are summarized in Table 3.2. The optimal solution was obtained using the CPlex solver by ILOG® and is shown in Table 3.3. In this case, the decision maker should include receiving antennas 1 and 2, transmitting antenna 2 and transponder 2 with a maximum reward of 68 units.

Table 3.1 Payload data for the single-launch problem with certain budget.

Index (k)	Payload	Reward	Cost	Wgt (lb)	Pow (W)	Vol (ft^3)
1	Rec Antenna 1	15	\$400k	200	350	3
2	Rec Antenna 2	10	\$200k	100	450	2
3	Tra Antenna 1	8	\$600k	400	300	4
4	Tra Antenna 2	18	\$500k	300	500	4
5	Transponder 1	23	\$800k	900	750	9
6	Transponder 2	25	\$950k	800	700	7

Table 3.2 Resources for the single-launch problem with certain budget.

Resource	Budget	Weight (lb)	Power (W)	Volume (ft^3)
Capacity	\$2.5 M	1,900	2,600	20

Table 3.3 Optimal solution for the single-launch problem with certain budget.

Index (k)	Payload	x_k	Reward Accrued
1	Rec Antenna 1	1	15
2	Rec Antenna 2	1	10
3	Tra Antenna 1	0	0
4	Tra Antenna 2	1	18
5	Transponder 1	0	0
6	Transponder 2	1	25

3.3 Single-Launch Models with Uncertainty

3.3.1 Payload Prioritization

The single-launch payload selection model with certainty (3.1) simply determines the best set of payloads to include on the satellite and does not assign priorities to the payloads. It is possible that with one (small) budget a desirable payload would be excluded but under another (larger) budget, it would be selected for inclusion. On the other hand, it is also possible that no matter what the budget level, an extremely desirable payload will be selected under all budget scenarios. Finally, it is also possible that a payload is selected at the smaller budget level but not at a higher budget level. This can occur because at the lower budget level a more attractive payload (in terms of reward) cannot fit within the budget, but it can fit at the higher level. Because it is very possible that the budget will change, and there is no guarantee that the selection decision will be consistent across all budget scenarios, it is advantageous to generate a prioritized list of payloads for inclusion on the satellite. The decision maker can then select payloads from the list until the budget is exhausted.

We now relax the assumption that the budget is known. This approach is appropriate when one must make payload selections prior to knowing which budget scenario will be realized. In many cases, the budget is known only in the form of a distributional forecast. A common approach is to build weighted budget scenarios by requesting a possible range of values and a most likely value from a subject matter expert. One possible set of scenarios consists of a low budget scenario with probability 0.1, an average budget scenario with probability 0.8, and a high budget scenario with probability 0.1. On the other hand, if the budget is known in the form of a continuous probability distribution, we can use the extended Pearson-Tukey method. This method approximates a continuous distribution using three discrete points by assigning the 0.05 fractile a probability of 0.185, assigning the median

probability of 0.63, and assigning the 0.95 fractile probability of 0.185. It has been shown that the Pearson-Tukey performs well as an approximation for a wide range of probability distributions [4].

In this model, we maintain the assumption that the rewards are determined in advance and do not change and there are n candidate payloads under consideration for inclusion on the satellite bus. Let $I = \{1, 2, \ldots, n\}$ denote the set of priority levels. For instance, a payload which is assigned priority level 1 receives the highest priority on the list. The model will decide which payloads receive a priority level and which are eliminated from consideration, if any. For $k \in K$, $i \in I$ and $\omega \in \Omega$, the decision variable x_{ki}^{ω} assumes a value of 1 if payload k has priority level i and level i is funded under budget scenario ω and 0 otherwise. A summary of the additional problem data follows:

Additional Sets:

$\omega \in \Omega$	set of budget scenarios
$i \in I$	set of priority levels

Data:

b^{ω}	budget under scenario ω, $b^1 < b^2 < \cdots < b^m$
q^{ω}	probability of budget scenario ω, $\omega \in \Omega$

Decision Variables:

x_{ki}^{ω}	1 if payload k has priority level i and level i is funded under budget scenario ω; 0 otherwise

Boundary Conditions:

$$x_{k0}^{\omega} \equiv 1, \quad k \in K, \omega \in \Omega$$

$$x_{ki}^{0} \equiv 0, \quad k \in K, i \in I$$

The payload prioritization formulation follows:

$$\max_{x} \quad \sum_{\omega\in\Omega}\sum_{k\in K}\sum_{i\in I} q^{\omega} r_k x^{\omega}_{ki} \tag{3.2a}$$

$$\text{s.t.} \quad \sum_{k\in K}\sum_{i\in I} c_k x^{\omega}_{ki} \le b^{\omega}, \qquad \omega \in \Omega \tag{3.2b}$$

$$\sum_{k\in K}\sum_{i\in I} a_{kd} x^{\omega}_{ki} \le u_d, \qquad d \in D, \omega \in \Omega \tag{3.2c}$$

$$\sum_{k\in K} x^{\omega}_{ki} \le 1, \qquad i \in I, \omega \in \Omega \tag{3.2d}$$

$$\sum_{i\in I} x^{\omega}_{ki} \le 1, \qquad k \in K, \omega \in \Omega \tag{3.2e}$$

$$x^{\omega}_{ki} \le \sum_{v\in K, v\neq k} x^{\omega}_{v,i-1}, \qquad k \in K, i \in I, \omega \in \Omega \tag{3.2f}$$

$$x^{\omega-1}_{ki} \le x^{\omega}_{ki}, \qquad k \in K, i \in I, \omega \in \Omega \tag{3.2g}$$

$$x^{\omega}_{ki} \in \{0,1\}, \qquad k \in K, i \in I, \omega \in \Omega. \tag{3.2h}$$

The objective function (3.2a) is the expected reward obtained over all possible budget scenarios. Constraints (3.2b) limit the cost of including payloads on the satellite to each budget scenario ω. Constraints (3.2c) limit the inclusion of payloads under budget scenario ω to the available specifications of the satellite. Constraints (3.2d) and (3.2e) define the priority list; each included payload is assigned to at most one priority level, and each priority level is assigned at most one payload. Constraints (3.2f) allow priority level i to be funded under scenario ω only if the next higher priority level is funded under the same scenario. Constraints (3.2g) allow payload k to be assigned priority level i only if the payload is assigned to the priority level under the next largest budget scenario.

We now illustrate the single-launch payload prioritization model with uncertain budget via a numerical example. Recall from the telecommunications satellite example that $K = \{1, 2, 3, 4, 5, 6\}$ represents the set of available payloads and $D = \{1, 2, 3\}$ denotes the respective engineering specifications of weight, power and volume. We maintain the reward, cost, weight, power and volume data for the payloads as well as

Table 3.4 Budgets for the single-launch prioritization example.

Index (ω)	State	Budget (b^ω)	Probability (q^ω)
1	Low	\$2.0 M	0.2
2	Med	\$2.5 M	0.6
3	High	\$3.0 M	0.2

Table 3.5 Optimal solution for the single-launch prioritization example.

Priority (i)	Payload	Included Budgets	Expected Reward
1	Transponder 2	Low, Med, High	25.0
2	Rec Antenna 1	Low, Med, High	15.0
3	Rec Antenna 2	Low, Med, High	10.0
4	Tra Antenna 2	Med, High	14.4
5	Tra Antenna 1	High	1.6
6	Transponder 1	-	0.0

the resource capacities for the satellite from Tables 3.1 and 3.2, respectively. Assume there are three potential budget scenarios as depicted in Table 3.4. The optimal solution was obtained using the CPlex solver by ILOG® and is summarized in Table 3.5. The payloads are in the prioritized order as indicated in the first column; the third column (Included Budgets) indicates the budget scenario levels under which the payload is included on the satellite; the last column is the expected reward accrued for including each payload. For instance, Transmitting Antenna 2 receives priority level 4, is included under the medium and high budget levels and has an expected reward of $(0.2 \times 0) + (0.6 \times 18) + (0.2 \times 18) = 14.4$ reward units. In this case, the maximum expected reward is 66 units which is obtained by summing the Expected Reward column. The solution is interpreted as follows. If the low budget is realized, each of the first three items on the list are included on the satellite. If the medium budget level is realized, the fourth payload is included in addition to the three from the low budget scenario. Finally, if the high budget is realized, the first five payloads on the priority list are included on the satellite bus.

By including an uncertain budget in the payload selection problem, we capture reality: uncertain budgets are common because payload selection decisions must be

made before budgetary decisions are finalized. When the payload selection problem is solved assuming a known budget, the optimal payload selection decision for a different budget scenario is not likely to be optimal. The payload prioritization model generates a rank-ordered list of payloads as opposed to the on/off decision of the payload selection problem with certainty. While the priority list assigns payloads distinct priority levels, all the payloads which are included under the same budget scenario essentially receive the same priority level. For instance, in Table 3.5, the payloads assigned to priority levels 1-3 all effectively receive the same (highest) priority level because all three are funded under all budget scenarios. Not only is the list prioritized, but the priority rankings apply to all budget scenarios.

3.3.2 Payload Prioritization with Requirements

We now assume there is a finite number of mission requirements that must be accomplished by the constellation. These requirements are derived from the mission of the constellation as defined in the design process described in Chapter 2. Thus, each mission of the constellation is characterized using a set of requirements. We select the payloads for inclusion on a satellite bus which best satisfy these requirements. When the satellite is replenishing a pre-existing constellation, the degrading payloads already in orbit dictate the mission areas, and thus, requirements to be included on the next satellite launch. The mission requirements of the constellation that are the most in need become the requirements with the greatest reward. Given a predetermined launch time in the single-launch case, the degradation of the payloads already in the constellation (and therefore the rewards) are considered as known quantities.

We now shift the focus of the model from selecting payloads that best fit onto the satellite bus to selecting payloads that best accomplish the mission requirements of the constellation. Therefore, we alter the objective function to reflect this change. Let J denote the finite set of mission requirements and let K_j denote the subset of

payloads that satisfy requirement $j \in J$. We partition the set J into two disjoint subsets, J_S and J_M. Let J_S represent the subset of requirements that have a sole-source, i.e., the set of requirements that may be satisfied by a single payload. Let J_M denote the subset of multi-source requirements, i.e., the set of requirements that may be satisfied by multiple, distinct payloads. The reward r_j is accrued if sole-source requirement $j \in J_S$ is satisfied on the satellite launch. Let z_j^ω denote the number of payloads which satisfy multi-source requirement j that are included on the satellite under budget scenario ω and let $f_j(z_j^\omega)$ be a piecewise-linear function such that the slope of each segment is the the marginal reward for increasing the number of payloads that satisfy requirement j across the defined segment. Let β_{js} denote the slope of the sth segment in the piecewise linear function $f_j(\cdot)$. In this case, using a standard formulation technique, the first included payload would receive a reward equal to the slope of the segment having largest slope. Therefore, if $f_j(\cdot)$ is not concave, additional logical constraints need to be added to ensure that the rewards are properly allocated, i.e., the first payload included on the satellite bus must receive the reward equal to the slope of the first segment.

Figure 3.1 shows three possible reward functions for multi-source requirements. It is important to note that the horizontal axis of the graphs is the discrete number of payloads satisfying the requirement. Therefore, the function is evaluated at integral points and the lines are drawn in order to illustrate the concavity or convexity of the function. Figure 3.1(a) illustrates a typical concave piecewise-linear function where there is a diminishing reward for each additional payload that satisfies requirement 1. This could be the case on a telecommunications satellite where the addition of identical transmitting antennas increases the reward gained, but with each additional antenna, the reward decreases. Figure 3.1(b) shows the case where requirement 2 demands at least three payloads to perform the mission since there is no reward gained for including one or two payloads, but a reward is accrued once a third payload is selected. Since this is a convex function, additional constraints must be

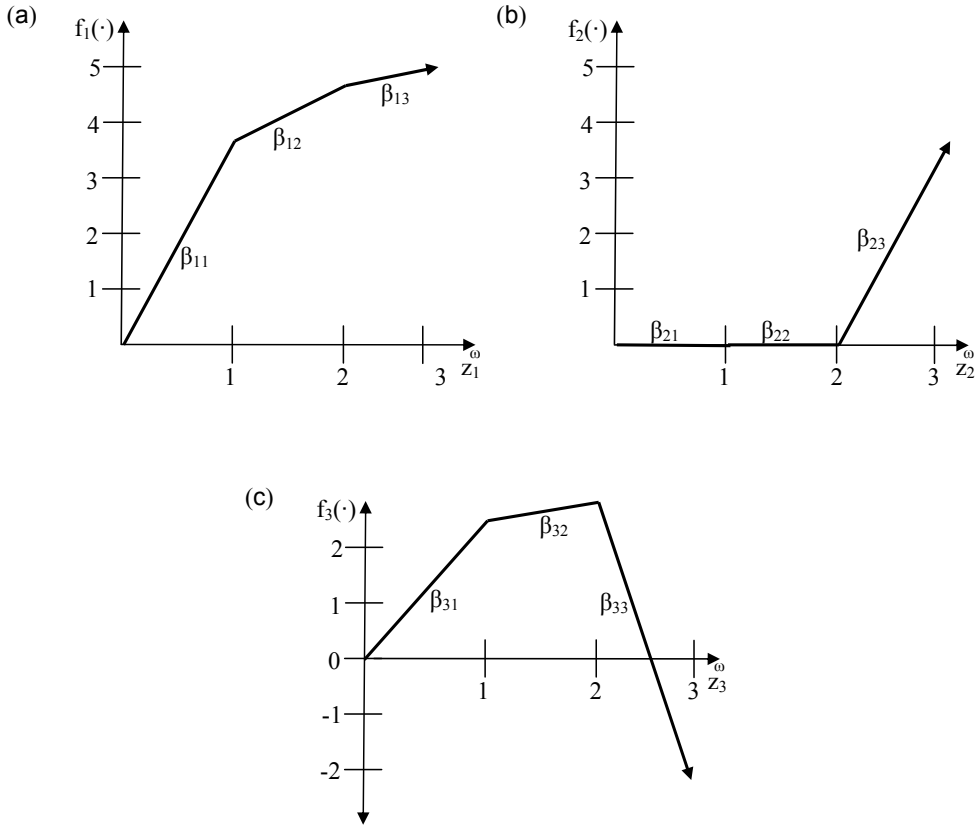

Figure 3.1 Piecewise-linear reward functions for multi-source requirements.

added to ensure the reward is properly allocated. This example might correspond to the case of imaging satellites from which images are required in three or more spectrums, one payload (camera) for each specific spectrum. Figure 3.1(c) represents the case where one or two payloads can perform the mission, but additional payloads cause the mission to fail. The negative slope for the third segment can be viewed as a penalty for having three payloads. This could be the case on a satellite where including two transmitting antennas does not allow signal interference. However, the addition of a third transmitting antenna causes enough signal interference so as to render all three antennas useless. Allowing $f(\cdot)$ to be either increasing or decreasing, and either concave or convex, makes the model more realistic since the reward function allows the decision maker to account for the diminishing rewards

for multiple identical payloads and to ensure that requirements are appropriately satisfied.

In this model, the objective is to maximize the reward obtained by satisfying the set of mission requirements. We assume that the rewards for sole-source requirements are constant and do not depend on the reward values of the other requirements. Likewise, the reward functions for the multi-source requirements are do not depend on the other reward functions and known with certainty. Let Ω represent the set of possible budget scenarios. Two decision variable types are introduced: for $j \in J_M$ and $\omega \in \Omega$ the decision variable z_j^ω is as defined above and for $j \in J_S$, $k \in K_j$ and $\omega \in \Omega$ the decision variable y_{kj}^ω assumes a value of 1 if payload k satisfies requirement j under budget scenario ω and 0 otherwise. A summary of the additional model data follows:

Additional Data:

r_j	reward received for satisfying sole-source requirement j
$f_j(\cdot)$	piecewise-linear reward function for satisfying multi-source requirement j

Decision Variables:

x_{ki}^ω	1 if payload k has priority level i and level i is funded under budget scenario ω; 0 otherwise
y_{kj}^ω	1 if payload k satisfies sole-source requirement j under budget scenario ω; 0 otherwise
z_j^ω	number of payloads which satisfy multi-source requirement j under budget scenario ω

The revised mathematical programming model follows:

$$\max_{x,y,z} \quad \sum_{\omega\in\Omega} q^{\omega}\left(\sum_{j\in J_S}\sum_{k\in K_j} r_j y_{kj}^{\omega} + \sum_{j\in J_M} f_j(z_j^{\omega})\right) \tag{3.3a}$$

$$\text{s.t.} \quad \sum_{k\in K}\sum_{i\in I} c_k x_{ki}^{\omega} \le b^{\omega}, \qquad \omega\in\Omega \tag{3.3b}$$

$$\sum_{k\in K}\sum_{i\in I} a_{kd} x_{ki}^{\omega} \le u_d, \qquad d\in D, \omega\in\Omega \tag{3.3c}$$

$$\sum_{k\in K} x_{ki}^{\omega} \le 1, \qquad i\in I, \omega\in\Omega \tag{3.3d}$$

$$\sum_{i\in I} x_{ki}^{\omega} \le 1, \qquad k\in K, \omega\in\Omega \tag{3.3e}$$

$$x_{ki}^{\omega} \le \sum_{v\in K, v\neq k} x_{v,i-1}^{\omega}, \qquad k\in K, i\in I, \omega\in\Omega \tag{3.3f}$$

$$x_{ki}^{\omega-1} \le x_{ki}^{\omega}, \qquad k\in K, i\in I, \omega\in\Omega \tag{3.3g}$$

$$\sum_{k\in K_j}\sum_{i\in I} x_{ki}^{\omega} \le 1, \qquad j\in J_S, \omega\in\Omega \tag{3.3h}$$

$$y_{kj}^{\omega} \le \sum_{i\in I} x_{ki}^{\omega}, \qquad j\in J_S, k\in K_j, \omega\in\Omega \tag{3.3i}$$

$$z_j^{\omega} = \sum_{k\in K_j}\sum_{i\in I} x_{ki}^{\omega}, \qquad j\in J_M, \omega\in\Omega \tag{3.3j}$$

$$x_{ki}^{\omega} \in \{0,1\}, \qquad k\in K, i\in I, \omega\in\Omega \tag{3.3k}$$

$$y_{kj}^{\omega} \in \{0,1\}, \qquad j\in J_S, k\in K_j, \omega\in\Omega \tag{3.3l}$$

$$z_j^{\omega} \in \mathbb{Z}_+, \qquad j\in J_M, \omega\in\Omega. \tag{3.3m}$$

The objective function (3.3a) is the expected reward for satisfying both sole- and multi-source requirements obtained over all possible budget scenarios. Constraints (3.3b) - (3.3g) are the same as those of Formulation (3.2) and are not affected by the adding of requirements to the model. Constraints (3.3h) ensure each sole-source requirement is satisfied by at most one payload. Constraints (3.3i) allow payload k to serve sole-source requirement j only if k is selected under budget scenario ω.

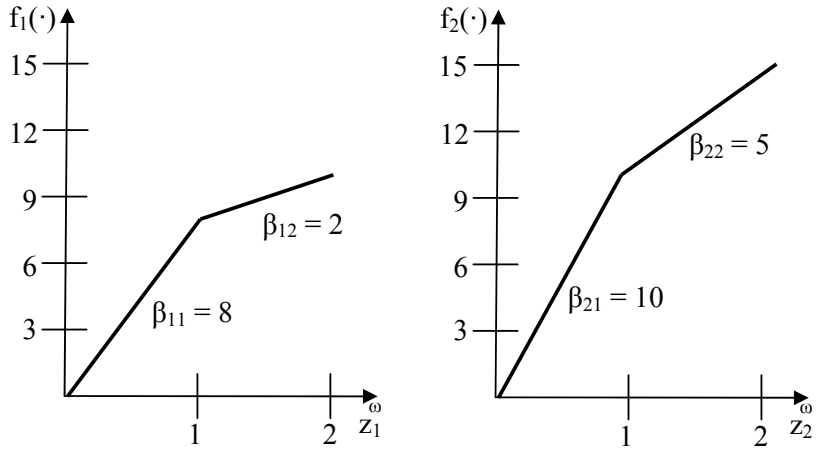

Figure 3.2 Reward functions for the prioritization problem with requirements.

Constraints (3.3j) count the number of payloads that satisfy multi-source requirement j under budget scenario ω. Constraints (3.3m) limit the number of payloads which satisfy multi-source requirements to being a non-negative integer. (Note this constraint is ensured to hold given (3.3j) and (3.3k).)

We continue our telecommunications satellite example to illustrate the single-launch payload prioritization model with requirements and an uncertain budget. Let $J = \{1, 2, 3\}$ denote the set of requirements as defined in Table 3.6. Let $K = \{1, 2, 3, 4, 5, 6\}$ denote the set of available payloads which are listed in Table 3.7. Table 3.8 shows the capacities of each of the engineering specifications denoted $D = \{1, 2, 3\}$. Table 3.9 depicts the budget scenarios under consideration. The reward functions for the multi-source requirements of receiving and transmitting antennas are given in Figure 3.2. The reward for satisfying the sole-source requirement of including a transponder capability (r_3) is 14. The optimal solution was obtained using CPlex solver by ILOG® and is shown in Table 3.10. The payloads are in the prioritized order as indicated in the first column; the third column is the requirement which is satisfied by including the payload; the fourth column (Included Budgets) indicates the budget scenario levels under which the payload is included on the satellite; the last column is the expected reward accrued for each payload.

Table 3.6 Requirements for the prioritization problem with requirements.

Index (j)	Requirement	Type (J_S or J_M)	Satisfying Payloads (K_j)
1	Receiving	Multi-Source	Rc Ant 1, Rc Ant 2
2	Transmitting	Multi-Source	Tr Ant 1, Tr Ant 2
3	Transponding	Sole-Source	Transp 1, Transp 2

Table 3.7 Payload data for the prioritization problem with requirements.

Index (k)	Payload	Cost	Wgt (lb)	Pow (W)	Vol (ft^3)
1	Rec Antenna 1	\$400k	200	350	3
2	Rec Antenna 2	\$200k	100	450	2
3	Tra Antenna 1	\$600k	400	300	4
4	Tra Antenna 2	\$500k	300	500	4
5	Transponder 1	\$800k	900	750	9
6	Transponder 2	\$950k	800	700	7

Table 3.8 Resources for the prioritization problem with requirements.

Index (d)	Resource	Capacity (u_d)
1	Weight (lb)	1,900
2	Power (W)	2,600
3	Volume (ft^3)	22

Table 3.9 Budgets for the prioritization problem with requirements.

Index (ω)	State	Budget (b^ω)	Probability (q^ω)
1	Low	\$1.5 M	0.2
2	Med	\$2.0 M	0.6
3	High	\$2.5 M	0.2

Table 3.10 Optimal solution for the prioritization problem with requirements.

Priority (i)	Payload	Satisfied Req	Included Budgets	Exp Reward
1	Transponder 1	Transponding	Low, Med, High	14.0
2	Rec Antenna 2	Receiving	Low, Med, High	8.0
3	Tra Antenna 2	Transmitting	Low, Med, High	10.0
4	Rec Antenna 1	Receiving	Med, High	1.6
5	Tra Antenna 1	Transmitting	High	1.0
6	Transponder 2	-	-	0.0

In this case, the maximum expected reward is 34.6 units which is calculated by summing the Expected Reward column. The solution is interpreted in the same manner as before. If the low budget is realized, each of the first three items on the list are included on the satellite bus. If the medium budget level is realized, the fourth payload is also included. Finally, if the high budget is realized, all five payloads on the priority list are included on the satellite bus.

By including requirements in the payload selection problem, we link the mission for which the constellation is designed to the decision of which payloads to include on a satellite bus. This approach allows greater flexibility in assigning payloads. In the payload selection problem with requirements, new or updated missions can easily be added to the requirements of the constellation. The new mission is incorporated by making the reward for satisfying the new requirement much larger than any other requirement. The partitioning of the requirements into sole- and multi-source requirements allows assigning payloads to the satellite in a more realistic manner.

3.4 Multiple-Launch Models

While the models of Section 3.3 account for uncertain budgets and mission requirements, they consider only a single satellite launch in which the reward values are time-invariant. Consider now a sequence of launches which will populate or replenish a satellite constellation in a finite time horizon. At fixed time epochs, a satellite will be launched into orbit to help satisfy the constellation's mission requirements. Satellite mission planners must take into account the composition of the constellation so that mission requirements are not neglected. Moreover, ignoring the composition of the constellation can lead to large cost overruns due to excessive payload redundancies. While redundancy can increase the probability of meeting the mission requirement, launching too many payloads is incredibly expensive. Therefore, reward values are tied to the constellation rather than the individual launches. The payloads to be included on each subsequent launch will take into account the

payloads which have been launched on previous satellites and are still functioning in the constellation at the time of the current launch. Consider a sole-source mission requirement currently satisfied in the constellation. A reward is accrued for meeting the requirement but no additional payloads are launched since we assume that only a single payload satisfies these types of requirements. In the case of multi-source mission requirements, the constellation receives diminishing rewards for the launch of additional payloads. Meanwhile, payloads which meet the requirements not already satisfied in the constellation will be given larger reward values.

Figure (3.3) shows a graphical depiction of a time line for a sequence of launches. Launch 1 occurs at time 0 and the satellite (and thus the payloads included on the satellite) has a constant mean mission life. In this illustrative example, the mean mission life is 3Δ time units, where Δ represents the constant inter-launch time. The first satellite functions until immediately before the fourth launch, at which time the satellite is considered to no longer be functioning. In a similar fashion, subsequent satellites are launched and operate for 3Δ time units before being considered failed. At the time of the current launch, the constellation is comprised of the payloads included on the two previous satellites launched.

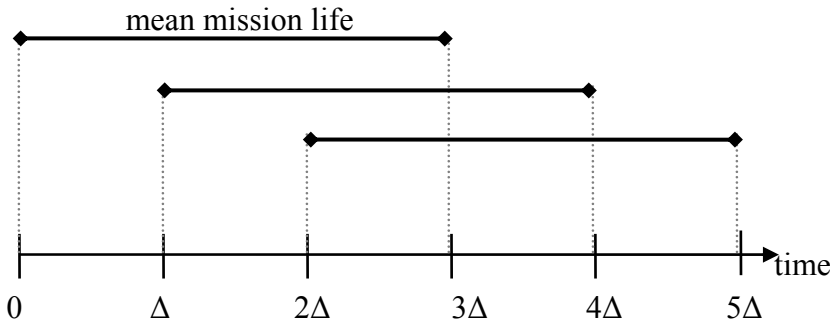

Figure 3.3 Time line for sequential satellite launches.

3.4.1 Sequential Payload Selection

We first consider a model in which the single-launch model is solved at each launch epoch and rewards are based on the composition of the constellation. We modify the single-launch prioritization model to account for the launch being considered and the composition of the constellation. This model is appropriate if the satellite program is subject to termination without advanced notice since we maximize the reward accrued on the current launch. In other words, if the program was cancelled, we would lose any reward that would otherwise have been accrued by waiting to include the payload on a subsequent launch.

Let L denote the set of launches in a finite time horizon where $l \in L$ denotes the launch currently under consideration. In order to account for the payloads currently in the constellation, we introduce two inventory decision variable types. Let IY_{jl}^{ω} indicate if sole-source requirement j is satisfied in the constellation at the time of launch l under budget scenario ω, and let IZ_{jl}^{ω} be the number of payloads which satisfy multi-source requirement j in the constellation at the time of launch l under budget scenario ω. We define the set W_l as the set of launch indices whose payloads are currently operating in the constellation. Recall that t is the mean mission life in number of launches. Thus, we define $W_l = \{l - t + 1, l - t + 2, \ldots, l\}$. For instance, if the current launch is launch 4 and the mean mission life is 3 launches, $W_4 = \{2, 3, 4\}$. The payloads included on launches two through four are functioning in the constellation and included in the calculation of the reward values. The constellation inventory variables are defined as follows:

$$IY_{jl}^{\omega} \leq \sum_{w \in W} \sum_{k \in K_j} y_{kjw}^{\omega}, \tag{3.4}$$

$$IZ_{jl}^{\omega} = \sum_{w \in W} z_{jw}^{\omega}, \tag{3.5}$$

where IY is binary and IZ is a nonnegative integer. We can account for the initial status of the constellation by assigning values for negative l values. For instance, $y^{\omega}_{k,j,-1} = 1$ implies that payload k satisfies sole-source requirement j in the constellation under budget scenario ω and has a remaining lifetime of $t - 2$ launches.

In the sequential multiple-launch model, a single-launch prioritization model is solved for each launch sequentially. The solutions from the previous launches are used to account for the composition of the constellation. We assume that sole-source requirements can be satisfied only once in the constellation and that multi-source requirements can be satisfied by more than one payload from more than one satellite in the constellation. A summary of the problem data follows.

Sets:

$k \in K$	set of candidate payloads (capabilities)
$d \in D$	set of included engineering specifications (weight, power, volume, etc.)
$\omega \in \Omega$	set of budget scenarios
$i \in I$	set of priority levels
$l \in L$	set of launches

Data:

c_{kl}	unit cost of payload k on launch l
a_{kd}	consumption of resource d by payload k
u_{dl}	capacity of resource d on launch l
b^{ω}_{l}	budget under scenario ω on launch l, $b^1_l < b^2_l < \cdots < b^m_l$
q^{ω}	probability of budget scenario ω, $\omega \in \Omega$
r_{jl}	reward received for satisfying sole-source requirement j on launch l
$f_{jl}(\cdot)$	piecewise-linear reward function for satisfying multi-source requirement j on launch l

Decision Variables:

x_{kil}^{ω}	1 if payload k has priority level i and level i is funded under budget scenario ω on launch l; 0 otherwise
y_{kjl}^{ω}	1 if payload k satisfies sole-source requirement j under budget scenario ω on launch l; 0 otherwise
z_{jl}^{ω}	number of payloads which satisfy multi-source requirement j under budget scenario ω on launch l
IY_{jl}^{ω}	1 if sole-source requirement j is satisfied in the constellation at the time of launch l under budget scenario ω; 0 otherwise
IZ_{jl}^{ω}	number of payloads satisfying multi-source requirement j in the constellation at the time of launch l under budget scenario ω

Boundary Conditions:

$x_{k0l}^{\omega} \equiv 1, \quad k \in K, l \in L, \omega \in \Omega$

$x_{kil}^{0} \equiv 0, \quad k \in K, i \in I, l \in L$

The revised mathematical programming model for launch l is as follows.

$$\max_{x,y,z,IY,IZ} \quad \sum_{\omega\in\Omega} q^{\omega}\left(\sum_{j\in J_S}\sum_{k\in K_j} IY^{\omega}_{jl} r_{jl} + \sum_{j\in J_M} f_{jl}(IZ^{\omega}_{jl})\right) \tag{3.6a}$$

$$\text{s.t.} \quad \sum_{k\in K}\sum_{i\in I} c_{kl} x^{\omega}_{kil} \le b^{\omega}_{l}, \qquad \omega\in\Omega \tag{3.6b}$$

$$\sum_{k\in K}\sum_{i\in I} a_{kd} x^{\omega}_{kil} \le u_{dl}, \qquad d\in D, \omega\in\Omega \tag{3.6c}$$

$$\sum_{k\in K} x^{\omega}_{kil} \le 1, \qquad i\in I, \omega\in\Omega \tag{3.6d}$$

$$\sum_{i\in I} x^{\omega}_{kil} \le 1, \qquad k\in K, \omega\in\Omega \tag{3.6e}$$

$$x^{\omega}_{kil} \le \sum_{v\in K, v\neq k} x^{\omega}_{v,i-1,l}, \qquad k\in K, i\in I, \omega\in\Omega \tag{3.6f}$$

$$x^{\omega-1}_{kil} \le x^{\omega}_{kil}, \qquad k\in K, i\in I, \omega\in\Omega \tag{3.6g}$$

$$\sum_{k\in K_j}\sum_{i\in I} x^{\omega}_{kil} \le 1, \qquad j\in J_S, \omega\in\Omega \tag{3.6h}$$

$$y^{\omega}_{kjl} = \sum_{i\in I} x^{\omega}_{kil}, \qquad j\in J_S, k\in K_j, \omega\in\Omega \tag{3.6i}$$

$$IY^{\omega}_{jl} \le \sum_{w\in W}\sum_{k\in K_j} y^{\omega}_{kjw}, \qquad j\in J_S, \omega\in\Omega \tag{3.6j}$$

$$z^{\omega}_{jl} = \sum_{k\in K_j}\sum_{i\in I} x^{\omega}_{kil}, \qquad j\in J_M, \omega\in\Omega \tag{3.6k}$$

$$IZ^{\omega}_{jl} = \sum_{w\in W} z^{\omega}_{jw}, \qquad j\in J_M, \omega\in\Omega \tag{3.6l}$$

$$x^{\omega}_{kil} \in \{0,1\}, \qquad k\in K, i\in I, \omega\in\Omega \tag{3.6m}$$

$$y^{\omega}_{kjl}, IY^{\omega}_{jl} \in \{0,1\}, \qquad j\in J_S, k\in K_j, \omega\in\Omega \tag{3.6n}$$

$$z^{\omega}_{jl}, IZ^{\omega}_{jl} \in \mathbb{Z}_{+}, \qquad j\in J_M, \omega\in\Omega \tag{3.6o}$$

The objective function (3.6a) is the expected reward obtained for satisfying both types of requirements over all possible budget scenarios on launch l. The total maximum expected reward for the entire launch horizon is computed by summing the

maximum expected reward for each launch. Constraints (3.6b) limit the cost of including payloads on the satellite of launch l to each budget scenario ω. Constraints (3.6c) limit the inclusion of payloads under budget scenario ω to the available resources of the satellite for the launch. Constraints (3.6d) and (3.6e) ensure each payload is assigned at most one priority level, and each priority level is assigned at most one payload under each budget scenario. Constraints (3.6f) allow priority level i to be funded under scenario ω only if the next higher priority level is funded under the same scenario for launch l with the boundary condition that $x^{\omega}_{k0l} \equiv 1$ for $k \in K$ and $\omega \in \Omega$. Constraints (3.6g) allow payload k to be assigned priority level i only if the payload is assigned to the priority level under the next largest budget scenario with the boundary condition that $x^{0}_{kil} \equiv 0$ for $k \in K$ and $i \in I$. Constraints (3.6h) ensure each sole-source requirement is satisfied by at most one payload for the launch. Constraints (3.6i) allow payload k to serve sole-source requirement j only if k is selected under budget scenario ω for that launch. Constraints (3.6j) indicate whether sole-source requirement j is satisfied in the constellation at the time of launch l. Constraints (3.6k) count the number of payloads that satisfy multi-source requirement j under budget scenario ω on the launch. Constraints (3.6l) count the number of payloads satisfying multi-source requirement j in the constellation at the time of launch l.

3.4.2 Payload Prioritization Considering the Future

In the previous subsection, we sequentially selected the best set of payloads to include on the satellite bus for each launch. The use of this approach ensures that the reward accrued for each launch is maximized, irrespective of the future launches in the launch horizon. In practice, however, planners do not ignore future satellite launches. They consider the benefit of excluding a payload on the current launch in order to better meet mission requirements by including the payload on some future

launch. Likewise, a payload may be included on the current launch if the reward for including it on subsequent launches decreases dramatically.

We now consider a model which solves the entire sequence of launches simultaneously. We extend the sequential multiple-launch model, Model (3.6), to accomplish this. The revised mathematical programming model is as follows.

$$\max_{x,y,z,IY,IZ} \quad \sum_{\omega\in\Omega} q^{\omega} \sum_{l\in L} \left(\sum_{j\in J_S} \sum_{k\in K_j} IY^{\omega}_{jl} r_{jl} + \sum_{j\in J_M} f_{jl}(IZ^{\omega}_{jl}) \right) \tag{3.7a}$$

$$\text{s.t.} \quad \sum_{k\in K} \sum_{i\in I} c_{kl} x^{\omega}_{kil} \le b^{\omega}_{l}, \qquad l \in L, \omega \in \Omega \tag{3.7b}$$

$$\sum_{k\in K} \sum_{i\in I} a_{kd} x^{\omega}_{kil} \le u_{dl}, \qquad d \in D, l \in L, \omega \in \Omega \tag{3.7c}$$

$$\sum_{k\in K} x^{\omega}_{kil} \le 1, \qquad i \in I, l \in L, \omega \in \Omega \tag{3.7d}$$

$$\sum_{i\in I} x^{\omega}_{kil} \le 1, \qquad k \in K, l \in L, \omega \in \Omega \tag{3.7e}$$

$$x^{\omega}_{kil} \le \sum_{v\in K, v\neq k} x^{\omega}_{v,i-1,l}, \qquad k \in K, i \in I, l \in L, \omega \in \Omega \tag{3.7f}$$

$$x^{\omega-1}_{kil} \le x^{\omega}_{kil}, \qquad k \in K, i \in I, l \in L, \omega \in \Omega \tag{3.7g}$$

$$\sum_{k\in K_j} \sum_{i\in I} x^{\omega}_{kil} \le 1, \qquad j \in J_S, l \in L, \omega \in \Omega \tag{3.7h}$$

$$y^{\omega}_{kjl} = \sum_{i\in I} x^{\omega}_{kil}, \qquad j \in J_S, k \in K_j, l \in L, \omega \in \Omega \tag{3.7i}$$

$$IY^{\omega}_{jl} \le \sum_{w\in W} \sum_{k\in K_j} y^{\omega}_{kjw}, \qquad j \in J_S, l \in L, \omega \in \Omega \tag{3.7j}$$

$$z^{\omega}_{jl} = \sum_{k\in K_j} \sum_{i\in I} x^{\omega}_{kil}, \qquad j \in J_M, l \in L, \omega \in \Omega \tag{3.7k}$$

$$IZ^{\omega}_{jl} = \sum_{w\in W} z^{\omega}_{jw}, \qquad j \in J_M, l \in L, \omega \in \Omega \tag{3.7l}$$

$$x^{\omega}_{kil} \in \{0,1\}, \qquad k \in K, i \in I, l \in L, \omega \in \Omega \tag{3.7m}$$

$$y^{\omega}_{kjl}, IY^{\omega}_{jl} \in \{0,1\}, \qquad j \in J_S, k \in K_j, l \in L, \omega \in \Omega \tag{3.7n}$$

$$z^{\omega}_{jl}, IZ^{\omega}_{jl} \in \mathbb{Z}_{+}, \qquad j \in J_M, l \in L, \omega \in \Omega \tag{3.7o}$$

The objective function (3.7a) is the expected reward obtained for satisfying both types of requirements over all possible budget scenarios for all launches. Constraints

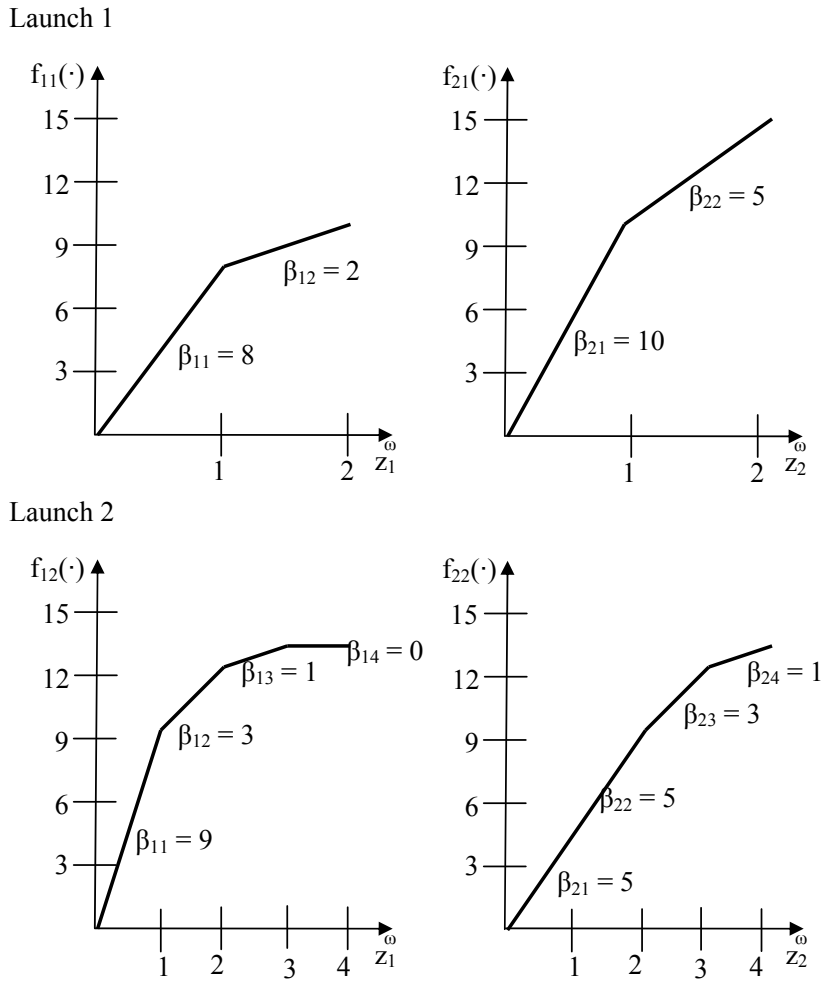

Figure 3.4 Reward functions for the memoryless prioritization problem.

(3.7b) - (3.7o) are the same as those of the previous formulation with the exception that the index l now spans all launches.

We illustrate the multiple-launch payload prioritization model using our telecommunications satellite example. Let $L = \{1, 2\}$ denote the set of two launches in the sequence and let $J = \{1, 2, 3\}$ represent the set of mission requirements which are defined in Table 3.6. Let $K = \{1, 2, 3, 4, 5, 6\}$ denote the set of available payloads; Table 3.11 lists these payloads and their respective specifications. The cost of each payload on each launch is depicted in Table 3.12. Let $D = \{1, 2, 3\}$ denote the set of engineering specifications whose capacities are listed in Table 3.13. Table 3.14

lists budget scenarios for each launch. The reward functions for the multi-source requirements of receiving and transmitting antennas for each launch are shown in Figure 3.4. The reward functions for launch 2 have four segments so rewards can be accrued for satisfying the requirement in the constellation, i.e., there is a reward value for the maximum number of payloads that can satisfy the requirement in the constellation. The rewards for satisfying the sole-source requirement of including a transponder capability on launch 1 is $r_{31} = 14$ and $r_{32} = 21$ for launch 2. The optimal solution was obtained using the CPlex solver by ILOG® and is shown in Table 3.15. The payloads are in the prioritized order as indicated in the first column; the third column (Included Budgets) indicates the budget scenario levels under which the payload is included on the satellite. The solution is interpreted as follows. If the low budget is realized, each of the first three items on the list are included on the satellite bus of launch 1 and the first two payloads are included on launch 2. If the medium budget level is realized, the fourth payload is also included on the first launch and the third payload is included on the second. Finally, if the high budget is realized, all five payloads on the priority list are included on the first satellite bus while no additional payloads are included on the final launch. There is no transponding capability included on the second launch since it is still functioning in the constellation as a result of being included on the launch 1. In this case, the maximum expected reward is 81.8 units. Rewards are accrued on launch 2 for satisfying requirements in the constellation in addition to the rewards accrued for including payloads on the satellite bus.

Table 3.11 Payload data for the memoryless prioritization example.

Index (k)	Payload	Wgt (lb)	Pow (W)	Vol (ft^3)
1	Rec Antenna 1	200	350	3
2	Rec Antenna 2	100	450	2
3	Tra Antenna 1	400	300	4
4	Tra Antenna 2	300	500	4
5	Transponder 1	900	750	9
6	Transponder 2	800	700	7

Table 3.12 Payload costs for the memoryless prioritization example.

Index (k)	Payload	Launch 1	Launch 2
1	Rec Antenna 1	\$400k	\$400k
2	Rec Antenna 2	\$200k	\$300k
3	Tra Antenna 1	\$600k	\$700k
4	Tra Antenna 2	\$500k	\$600k
5	Transponder 1	\$800k	\$800k
6	Transponder 2	\$950k	\$900k

Table 3.13 Resources for the memoryless prioritization example.

Index (d)	Resource	Launch 1	Launch 2
1	Weight (lb)	1,900	1,500
2	Power (W)	2,600	2,300
3	Volume (ft^3)	22	18

Table 3.14 Budgets for the memoryless prioritization example.

Index (ω)	State	Probability (q^ω)	Launch 1	Launch 2
1	Low	0.2	\$1.5 M	\$1.2 M
2	Med	0.6	\$2.0 M	\$1.6 M
3	High	0.2	\$2.5 M	\$2.0 M

Table 3.15 Solution for memoryless prioritization example, (a) launch 1; (b) launch 2.

Priority (i)	Payload	Satisfied Req	Included Budgets
(a)			
1	Transponder 1	Transponding	Low, Med, High
2	Tra Antenna 2	Transmitting	Low, Med, High
3	Rec Antenna 2	Receiving	Low, Med, High
4	Rec Antenna 1	Receiving	Med, High
5	Tra Antenna 1	Transmitting	High
6	Transponder 2	-	-
(b)			
1	Rec Antenna 2	Receiving	Low, Med, High
2	Tra Antenna 1	Transmitting	Low, Med, High
3	Tra Antenna 2	Transmitting	Med, High
4	Rec Antenna 1	-	-
5	Transponder 1	-	-
6	Transponder 2	-	-

In this chapter, we have developed optimization models for prioritizing payload launches which take into account uncertain budgets, mission requirements and constellation dependence. In the next chapter, we will evaluate the benefit of using these models by first comparing the single-launch prioritization model to a payload selection heuristic. We also evaluate the multiple-launch models to demonstrate the benefit of considering the future.

4. Computational Results

In this chapter, we compare the single-launch payload prioritization model to a greedy payload selection heuristic. Small, medium and large problem instances are randomly-generated, solved, and their objective function values are compared based on mean, median and maximum percent improvement. We also compare the multiple-launch prioritization models with and without consideration of the future. Various launch horizon and mean mission life values are used in randomly-generated problem instances to compare the optimal objective function values for each of the two types of multiple-launch models.

4.1 Overview of Experiments

In the single-launch experiments, we will demonstrate the advantages of using the prioritization model rather than a greedy payload selection heuristic. In the multiple-launch experiments, we demonstrate the benefit of considering the future in the multiple-launch prioritization models. All experiments were conducted on an IBM® Thinkpad with a 1.86 GHz Intel® Centrino processor and 0.99 GB of memory.

4.1.1 Single-Launch Overview

In order to analyze the results of the single-launch payload prioritization model, Model (3.3), we employ a greedy heuristic that might be used in a realistic payload selection strategy. The expected budget for the launch is first computed and used in a modified payload prioritization formulation, Model (3.3) with only a single budget scenario. The problem, solved using CPlex solver by ILOG®, yields the optimal reward for the expected budget and the list of payloads that should be included on the satellite bus. After a budget realization, we use one of two heuristics to adjust the payloads. In the first, we remove payloads from the list incrementally to drive the cost below the budgets which are less than the expected budget. In the

second, we add payloads to the list within the confines of the engineering specification constraints and ensuring the total cost remains less than the higher budget.

The following notation is used in both the low and high budget heuristics. Let R denote the optimal reward value obtained when the model is solved using the expected budget level, and let $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ denote the optimal solution with respective definitions,

$$x_k = \begin{cases} 1 & \text{if payload } k \text{ is included} \\ 0 & \text{otherwise} \end{cases},$$

$$y_{kj} = \begin{cases} 1 & \text{if payload } k \text{ satisfies sole-source requirement } j \\ 0 & \text{otherwise} \end{cases},$$

$$z_j = \text{the number of payloads that satisfy multi-source requirement } j.$$

Define $C = \sum_{k \in K} c_k x_k$ as the total cost of the payloads selected for inclusion on the satellite bus.

We first describe the low budget heuristic, i.e., that used to remove payloads from the solution of the mean-value problem, when the budget realization is lower than the mean. The objective of the heuristic is to remove payloads from the bus until C is less than B, the realized budget level. Define $\boldsymbol{m} = [m_1, m_2, \ldots, m_n]$ as the row vector of the marginal reward contributed by each payload to the total where n denotes the number of payloads being considered and m_k is the marginal reward contributed by payload k. If payload k satisfies a sole-source mission requirement j, the minimum contributed reward is equal to the reward for satisfying the requirement, i.e., $m_k = r_j$. Otherwise, if payload k satisfies multi-source mission requirement j, we use z_j, the number of included payloads which satisfy the requirement, from the solution to determine the marginal reward. The marginal reward contributed is equal to β_{j,z_j}. We select the smallest marginal reward, m_v; ties are broken by

Given: $\boldsymbol{x}$, $\boldsymbol{z}$, R, B, C, $c_k \; \forall \; k \in K$,
$r_j \; \forall \; j \in J_S$, $\beta_{j,z_j} \; \forall \; j \in J_M$
while $B < C$ **do**
for $x_k = 1, k \in K_j, j \in J_S$ **do**
$m_k \leftarrow r_j$
end for
for $x_k = 1, k \in K_j, j \in J_M$ **do**
$m_k \leftarrow \beta_{j,z_j}$
end for
Let: $v \leftarrow \text{argmin}\{m_k : x_k = 1\}$
$x_v \leftarrow 0$
$R \leftarrow R - m_v$
$C \leftarrow C - c_v$
end while

Figure 4.1 Low budget heuristic algorithm.

selecting the payload with the higher cost. To remove the payload from the satellite bus we set $x_v = 0$. Finally, we update the total cost, $C = C - c_v$, and the total reward, $R = R - m_v$. This process continues until the total cost of the payloads on the satellite bus is within the desired budget level. The low budget heuristic algorithm is outlined in Figure 4.1.

We now describe the high budget heuristic. The objective is to add payloads to the bus without violating the realized budget, B, or engineering specification constraints. We use H to denote the set of eligible payloads. If payload k satisfies a multi-source requirement, i.e., $k \in K_j$ and $j \in J_M$, then k is included in H if $x_k = 0$. If k satisfies a sole-source requirement, i.e., $k \in K_j$ and $j \in J_S$, then k is included in H only if $y_{kj} = 0$ for all $k \in K_j, j \in J_S$, i.e., no other payload is satisfying j. Let $\boldsymbol{m} = [m_1, m_2, \ldots, m_n]$ denote the marginal reward contributed by each eligible payload to the total where n denotes the number of payloads under consideration and m_k denotes the marginal reward contributed by payload k. If payload k satisfies a sole-source mission requirement j, the marginal reward is equal to the reward for satisfying the requirement, i.e., $m_k = r_j$. Otherwise, if payload k satisfies multi-source mission requirement j, we use z_j, the number of included payloads which

Given: $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$, R, B, C, $c_k \; \forall \; k \in K$, $r_j \; \forall \; j \in J_S$,
$\beta_{j,z_j} \; \forall \; j \in J_M$, $A_d \; \forall \; d \in D$, $a_{kd} \; \forall \; k \in K, d \in D$
Set: $H = \{k : x_k = 0, k \in K_j, j \in J_M\} \cup$
$\{k : y_{kj} = 0, k \in K_j, j \in J_S\}$
while $H \neq \emptyset$ **do**
  for $k \in H$ **do**
    if $k \in K_j, j \in J_S$ **then**
      $m_k \leftarrow r_j$
    else if $k \in K_j, j \in J_M$ **then**
      $m_k \leftarrow \beta_{j,z_j+1}$
    end if
  end for
  Let: $v \leftarrow \text{argmax}\{m_k : k \in H\}$
  if $B \geq C + c_v, A_d \geq \sum_k a_{kd} x_k + a_{vd}$ **then**
    $x_v \leftarrow 1$
    $H = H \setminus \{v\}$
    $R \leftarrow R + m_v$
    $C \leftarrow C + c_v$
  else
    $H = H \setminus \{v\}$
  end if
end while

Figure 4.2 High budget heuristic algorithm.

satisfy the requirement, from the solution to determine the marginal reward. The marginal reward contributed is β_{j,z_j+1}. We select the largest marginal reward, m_v. If the addition of payload v does not exceed the realized budget (i.e., if $C + c_v < B$) or the engineering specifications (i.e., if $\sum_k a_{kd} x_k + a_{vd} < A_d$), then payload v is added to the satellite bus, i.e., $x_v = 1$. We update the total cost, $C = C + c_v$, update the total reward, $R = R + m_v$, and remove payload v from the eligible payload list, $H = H \setminus \{v\}$. If the addition of payload v does violate the budget or engineering specification constraints, we remove payload v from the eligible list, $H = H \setminus \{v\}$. This process continues until all payloads are removed from the eligible list. The high budget heuristic algorithm is outlined in Figure 4.2.

We compute the expected reward of the payload selection heuristic using the weighted sum of the reward for each of the three budget scenarios. For instance, if

the low budget reward is 20, the medium reward 50 and the high reward 100 with respective probabilities 0.2, 0.6, and 0.2, the total expected reward is $(0.2 \times 20) + (0.6 \times 50) + (0.2 \times 100) = 54$ reward units. We treat the payload selection heuristic objective function value as a baseline and measure the percent improvement in the objective function value obtained by using the payload prioritization model. Let H^* denote the expected reward of the payload selection heuristic and let P^* denote the optimal expected reward obtained from the payload prioritization model. The percent improvement is given by

$$\left(\frac{P^*}{H^*} - 1\right) \times 100\%. \tag{4.1}$$

The solution to the payload selection heuristic will always yield a feasible solution to the single-launch prioritization model. However, the objective function value will only sometimes be optimal. Therefore, $P^* \geq H^*$, and we always see nonnegative percentage improvements. We compare the mean, median and maximum percent improvement in our numerical experiments.

One hundred problem instances were randomly-generated to compare the payload selection heuristic and the payload prioritization model. We consider three problem sizes: small (0 - 250 variables), medium (250 - 1,000 variables), and large (1,000+ variables). The problem size, number of payloads, sole-source requirements and multi-source requirements considered in this experiment are listed in Table 4.1. Payloads are divided evenly among the requirements, i.e., each mission requirement has the same number of payloads with which it can be satisfied. For instance, in the large problem there are three payloads for each mission requirement. The number

Table 4.1 Problem sizes for random problem instances.

Problem Size	# Pylds	S.S. Reqs	M.S. Reqs	Integer Vars	Binary Vars
Small	6	1	2	6	117
Medium	12	2	2	6	468
Large	18	2	4	12	1008

of integer and binary decision variables in the prioritization model for each case are listed in the last two columns.

The payload parameter distributions are summarized in Table 4.2. For example, each cost c_k is selected by drawing a realization from a continuous U(250,1000) distribution, independent of the other parameter values. The cost, weight, power and volume ranges are based on realistic payload information obtained from Wertz and Larson[24]. The sole-source rewards are assigned a continuous uniform value between 0 and 50. We assume that the reward functions for all multi-source requirements are concave. The multi-source requirements are assigned a continuous uniformly distributed initial reward on the range 0 to 50. Subsequent rewards are assigned uniformly between 0 and the previous reward to account for the diminishing reward accrued.

The satellite bus capacities are based on the randomly-generated payload data. Let C, W, P, and V denote the total cost, weight, power and volume of the randomly-generated payloads, respectively. The distributions of the satellite bus capacities are shown in Table 4.3. We bound the engineering specifications below by 25 percent of the total physical needs of the randomly-generated payloads, C. This ensures that

Table 4.2 Payload parameter distributions.

Parameter	Distribution
Cost ($100k)	U(250, 1,000)
Weight (lbs)	U(50, 1,000)
Power (W)	U(50, 1,000)
Volume (ft^3)	U(2, 15)
Sole-Source Reward	U(0, 50)
Multi-Source Reward	U(0, 50)

Table 4.3 Satellite bus capacity distributions.

Parameter	Distribution
Weight (lbs)	U($0.25W$, W)
Power (W)	U($0.25P$, P)
Volume (ft^3)	U($0.25V$, V)

there will be enough space on the satellite bus to include payloads. The specifications are bounded above by the total consumption of all the randomly-generated payloads. Randomly choosing the satellite bus capacities allows each problem instance to be constrained by at least one of the engineering specifications.

We assume that the budget for each launch is beta distributed with scale parameter α and shape parameter β, and the range is scaled to the total cost of the available payloads. We consider three possible parameter combinations for the continuous budget distributions. First, we set $\alpha = 1$ and $\beta = 1$ and scale the range of the distribution to cover the interval $[250, C]$. This is equivalent to using a continuous uniform distribution on the budget interval. For the right-skewed budget scenario, we use $\alpha = 1.5$ and $\beta = 3$ scaled to the same budget interval. Finally, we consider $\alpha = 5$ and $\beta = 5$ to obtain a truncated normal distribution. The three beta distributions are depicted over the range $[0, 1]$ in Figure 4.3. We use the extended Pearson-Tukey method[4] to approximate each of the three continuous distributions using three discrete points. The 0.05 and 0.95 fractiles are assigned a probability of 0.185 and the median is assigned probability 0.63. The MATLAB® and GAMS® codes for the single-launch experiments are included in Appendix A.

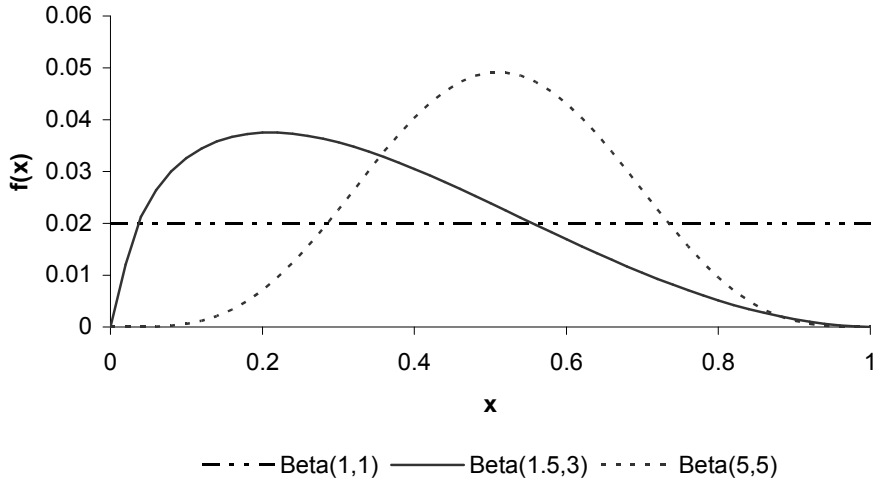

Figure 4.3 Budget distributions used for Pearson-Tukey approximations.

4.1.2 Multiple-Launch Overview

We compare the optimal objective function values of the sequential multiple-launch prioritization model and the multiple-launch prioritization model which considers the future to determine the benefit of accounting for future launches. We measure the percent improvement of the model considering the future over that of the sequential model. The optimal objective function value of the model considering future launches is denoted F^* and the optimal objective function value of the sequential model is denoted S^*. We compute the percent improvement by

$$\left(\frac{F^*}{S^*} - 1\right) \times 100\%. \tag{4.2}$$

It is worth noting that the model which considers the future will never yield a smaller objective function value than the sequential multiple-launch model since the latter always yields a feasible solution to the former. In other words, $F^* \geq S^*$. Thus, as in the single-launch experiment, we will always see nonnegative improvement.

Problem instances were generated for twelve potential payloads using the parameters listed in Tables 4.2 and 4.3. We consider only the right-skewed Pearson-Tukey approximation of the budget distribution. We use mean mission lives of 3, 4 and 5 launches and launch horizons of 5 and 8 launches. If $\Delta = 2$, the horizon is 10 or 16 years and the payload lifetimes are 6, 8 or 10 years, respectively. We assume the constellation is empty at the time of the first launch. Table 4.4 shows the problem size for each launch horizon; the sequential model and the model which considers the future have the same number of variables. The MATLAB® and GAMS® codes for the multiple-launch experiments are included in Appendix B.

Table 4.4 Problem sizes for random problem instances.

Pylds	Launches	S.S. Reqs	M.S. Reqs	Integer Vars	Binary Vars
12	5	2	2	120	2940
12	8	2	2	192	4704

4.2 Numerical Results and Conclusions

4.2.1 Single-Launch Experiments

We consider three cases: six payloads, 12 payloads and 18 payloads. In each case, 100 randomly-generated problem instances were solved by both the payload selection heuristic and the payload prioritization model. The percent improvement was calculated using the heuristic as the baseline value and the prioritization scheme as the improved solution. The mean, median, and maximum percent improvements are summarized in Table 4.5. If the heuristic yielded the optimal objective function value, the instance is counted in the 'No Improvement' column. The number of instances whose solutions lied within various ranges of improvement is provided in Table 4.6. Histograms for the percent improvement under each budget distribution approximation are shown in Figures 4.4 - 4.6. The areas shaded with upward lines (/) correspond to the uniformly distributed budget scenarios, the areas which are shaded in solid gray correspond to the right-skewed budget scenarios, and the areas shaded with downward lines (\) correspond to the truncated normal budget scenarios.

Table 4.5 Percent improvement for random, single-launch problem instances.

Pylds.	Bgt. Wght.	Mean	Median	Max
6	Uniform	2.863	2.695	15.670
	Right-skewed	7.376	7.907	81.420
	Truncated normal	1.712	2.415	9.500
12	Uniform	2.304	1.450	13.120
	Right-skewed	5.183	5.206	28.920
	Truncated normal	1.278	1.927	7.820
18	Uniform	2.142	3.182	10.080
	Right-skewed	4.568	3.860	17.970
	Truncated normal	0.644	1.127	3.520

Table 4.6 Number of single-launch solutions showing improvement.

Pylds.	Bgt. Wght.	No Imp.	0 - 5%	5 - 10%	>10%
6	Uniform	54	15	20	11
	Right-skewed	46	10	20	24
	Truncated normal	61	24	15	0
12	Uniform	46	29	22	3
	Right-skewed	27	30	24	19
	Truncated normal	43	53	4	0
18	Uniform	35	51	13	1
	Right-skewed	13	53	22	12
	Truncated normal	40	60	0	0

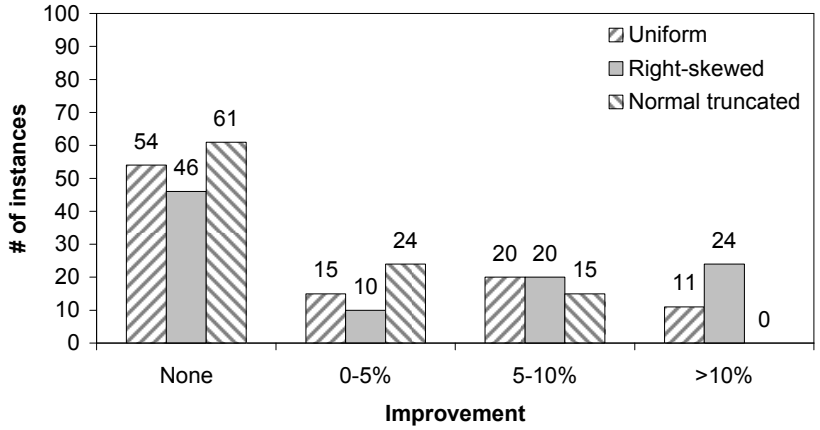

Figure 4.4 Histogram of percent improvements for small problem instances.

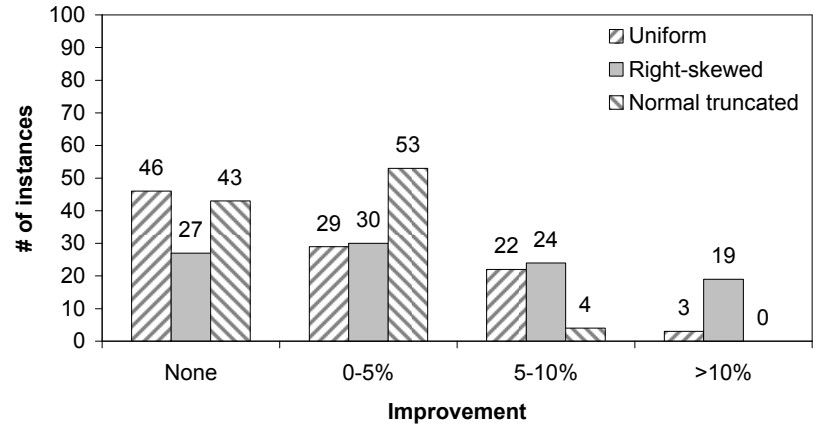

Figure 4.5 Histogram of percent improvements for medium problem instances.

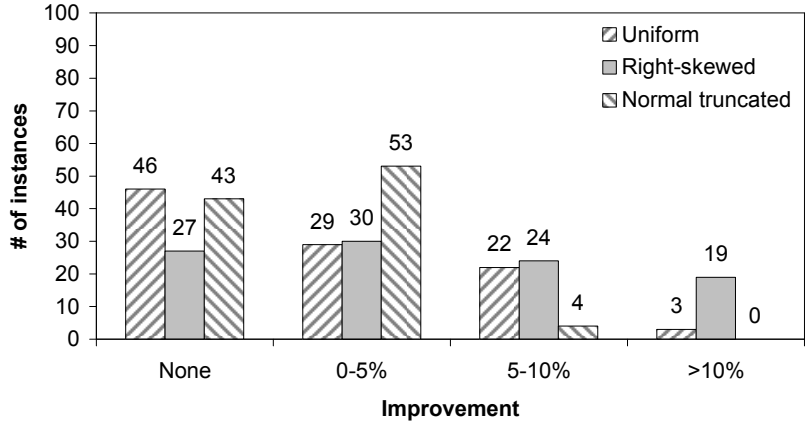

Figure 4.6 Histogram of percent improvements for large problem instances.

Overall, the payload prioritization model outperforms the payload selection heuristic. For the medium and large problem instances, the model had improvement over the heuristic greater than 50% of the instances. The payload prioritization model shows the most improvement when the budget distribution is skewed. In the presence of right-skewed budget scenarios, the model had a more considerable (> 5%) mean improvement than did the heuristic. The uniform and truncated normal budget scenarios also showed improvement, but to a lesser extent. Likewise, the number of problem instances in which the heuristic yielded the optimal solution was substantially smaller when the budget distribution was skewed to the left. A considerable number of problem instances yielded percent improvements greater than ten percent. In general, the experiments that assumed right-skewed budget scenarios appear to yield greater improvements than the other budget distributions.

4.2.2 Multiple-Launch Experiments

For the multiple-launch experiment, we consider two launch horizons and three payload mean mission lives. For every launch-lifetime combination, we solved the sequential payload prioritization model and the model which accounts for the future using 100 randomly-generated problem instances with twelve payloads each. We used the right-skewed budget scenario approximations as they showed the greatest

improvement for the single-launch model. The percent improvement was calculated using the sequential launch model as the baseline value and the model which considers the future as the improved solution. The mean, median, and maximum percent improvements are summarized in Table 4.7. If the sequential model yielded the optimal objective function value, the instance is counted in the 'No Improvement' column. The number of instances whose solutions lied within various ranges of improvement is provided in Table 4.8. Histograms for the percent improvement under each launch horizon for each mean mission life are shown in Figures 4.7 and 4.8. The areas shaded with upward lines (/) correspond to the experiments with a mean mission life of 3 launches, the areas which are shaded in solid gray correspond to a mean mission life of 4, and the areas shaded with downward lines (\) correspond to instances in which the mean mission life is 5.

Table 4.7 Percent improvement for random, single-launch problem instances.

Pylds.	Bgt. Wght.	Mean %	Median %	Max %	Min %
5	3	1.911	2.829	11.134	
	4	2.182	1.683	16.589	
	5	2.607	2.794	20.004	
8	3	1.879	2.334	6.918	
	4	1.774	1.650	7.283	
	5	2.139	2.374	10.060	

Table 4.8 Number of single-launch solutions showing improvement.

Pylds.	Bgt. Wght.	No Imp.	0 - 5%	5 - 10%	>10%
5	3	9	82	7	2
	4	6	82	8	4
	5	10	74	9	7
8	3	3	93	4	0
	4	3	92	5	0
	5	3	92	4	1

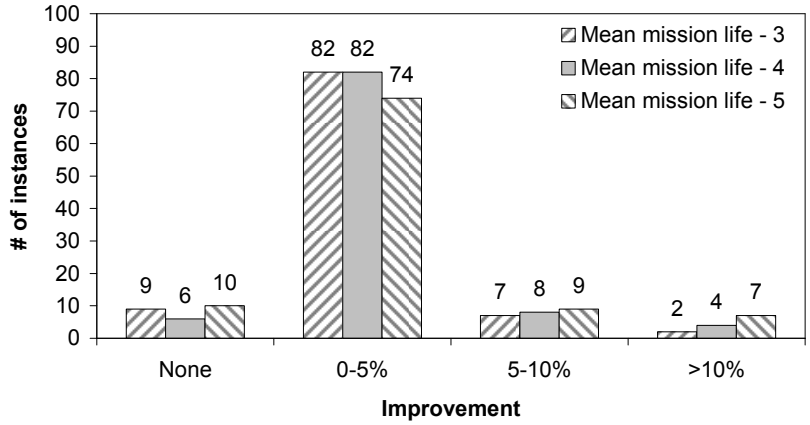

Figure 4.7 Histogram of percent improvements for 5-launch problem instances.

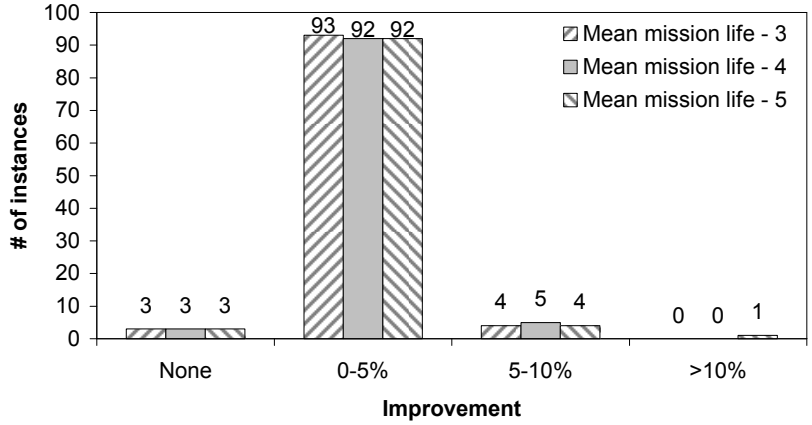

Figure 4.8 Histogram of percent improvements for 8-launch problem instances.

Overall, the model which considers the future outperforms the sequential model. The model which considers the future yielded an improved objective function value in over 90% of the problem instances for the five-launch horizon and 97% of the problem instances for the eight-launch horizon when considering a right-skewed budget distribution. While the mean percent improvement is relatively small, recall that the improvement is in addition to the larger percent improvements seen in the single-launch experiments over the same budget distribution.

The problem of selecting which payloads to include on a satellite is complex. When the budget distribution is symmetric for a single satellite launch, a simplistic greedy heuristic can be used for the selection process. However, when the budget dis-

tribution is skewed, the payload prioritization model shows substantial improvement over the selection heuristic. With skewed budget scenarios, incorporating multiple launches and consideration of future launches in the prioritization model further improves the objective function value over solving a sequence of launches without regard for the future. In the next chapter, we summarize the conclusions, offer some recommendations, and provide suggestions for future extensions of this work.

5. Conclusions and Future Research

We developed an optimization model which prioritizes the available payloads for each satellite bus in a sequence of launches such that the reward to the constellation is maximized. The models progressed in complexity from a simple, deterministic single-launch payload selection model to a complex, multiple-launch payload prioritization model which accounts for the constellation's mission requirements, an uncertain budget and a dependence on the composition of the constellation. We then compared the single-launch prioritization model to a greedy payload selection heuristic. We also compared two multiple-launch models in order to demonstrate the improvements gained when future launches are, or are not, considered. The models provided herein can be used in a variety of applications for which analysts have a need to prioritize item selections.

We first demonstrated that including priority levels in the formulation of a single-launch payload selection model improves the solution obtained by a greedy heuristic when uncertain budget scenarios prevail. The heuristic to which we compared the prioritization model used the optimal set of payloads under the expected budget. The heuristic removed payloads from the satellite to drive the cost below the realized budget or added payloads without exceeding the realized budget level while maintaining feasibility. When the budget scenario distribution is skewed, the single-launch payload prioritization model substantially outperforms the payload selection heuristic.

Next we extended the prioritization model to include multiple launches. Two formulations of this model were presented: the sequential prioritization model and the multiple-launch prioritization model which considers future launches. The difference in these models is that, in the sequential case, we do not account for payloads included on future launches. Whereas in the model that considers the future, we select payloads for all the launches in the time horizon simultaneously. In both models,

rewards are accrued for payloads existing in the constellation. When skewed budget scenario distributions are considered, the multiple-launch prioritization model that considers the future outperforms the model that does not.

Although the prioritization model captures the essence of the payload selection problem, there are many ways in which the methodology can be improved. We made several simplifying assumptions in our model: constant inter-launch times, constant payload mean mission lives, and temporal dependence. The introduction of uncertain inter-launch times would make the model more realistic in the sense that launch schedules are not fixed. In particular, manufacturing delays significantly add uncertainty that is not accounted for in the present models. Likewise, the mean mission life of each satellite payload was assumed to be constant. Incorporating a more realistic degradation process may prove worthwhile. Finally, we assumed that budget scenarios had temporal dependence. In reality, satellite launch budgets are independent and, therefore, so are their realized budget scenarios. Introducing temporal independence, i.e., each launch can realize a different budget scenario (e.g., high, medium, low), may improve the model. While each of these enhancements may drastically increase the complexity of the formulation, they may also increase realism of the prioritization models.

Bibliography

1. Bell, A.J. (2002). *Analysis of GPS Satellite Allocation for the United States Nuclear Detonation Detection System (USNDS).* M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH.

2. Birge, J.R. and F. Louveaux (1997). *Introduction to Stochastic Programming.* Springer-Verlag, New York, NY.

3. Brown, G., Dell R., Holtz H. and A. Newman (2003). How Space Command optimizes long term investment in space systems. *Interfaces*, **33**, (4), 1-14.

4. Clemen, R.T. and T. Reilly (2001). *Making Hard Decisions.* Duxbury, USA.

5. Dean, B.C., Goemans, M.X. and J. Vondrak (2004). Approximating the stochastic knapsack problem: The benefit of adaptivity. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, Rome, Italy, October 2004, 208-217.

6. Dean, B. C., Goemans, M.X. and J. Vondrak (2005). Adaptivity and approximation for stochastic packing problems. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, Vancouver, BC, Canada, January 2005, 395-404.

7. Diekelman, D. (1998). Design guidelines for post-2000 constellations. *Proceedings of Mission Design and Implementation of Satellite Constellations Workshop*, Toulouse, France, November 1997, 11-21.

8. EchoStar Communications Corporation. (2006). *2005 Annual Report.* Englewood, CO.

9. Farias, V.F. and B. Van Roy (2005). Approximation algorithms for dynamic resource allocation. *Operations Research Letters.* **34**, 180-190.

10. Flory, J.A. (2006). *Optimizing Mean Mission Duration for Multiple-Payload Satellites.* M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH.

11. Garey, M.R. and D.S. Johnson (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, CA.

12. General Accounting Office (2005). *Defense Acquisitions: Incentives and Pressures That Drive Problems Affecting Satellite and Related Acquisitions.* Washington D.C., GPO.

13. Golden, B.L., Levy, L. and R. Vohra (1987). The orienteering problem. *Naval Research Logistics*, **34**, 307-318.

14. Jacobs, J.L., Amato, A.J., Bolduc, M.D., Butler, S.P. and J.E. Caspero (1992). OSCARS - Operational Constellation Availability and Reliability Simulation. *Proceedings of the 1992 Summer Computer Simulation Conference*, Reno, Nevada, July 27-30, 878-882.

15. Jung, Ho-Won (1998). Optimizing value and cost in requirements analysis. *IEEE Software*, **15**, 4, 74-78.

16. Lansard, E. and J.-L. Palmade (1998). Satellite constellation design: searching for global cost-efficiency trade-offs. *Proceedings of Mission Design and Implementation of Satellite Constellations Workshop*, Toulouse, France, November 1997, 23-31.

17. Laporte, G. and S. Martello (1990). The selective travelling salesman problem. *Discrete Applied Mathematics*, **26**, 193-207.

18. Mettu, R.R. and G.C. Plaxton (2003). The online median problem. *SIAM Journal on Computing*, **32**, 2, 816-832.

19. Morton, D.P., Pan, F. and K.J. Saeger. (2007) Models for nuclear smuggling interdiction. *IIE Transactions*, **38**, 3-14.

20. Pan, F. and D.P. Morton. Minimizing a stochastic maximum-reliability path. *Networks*. To appear.

21. Simth, D.E. (2004). Choosing objectives in over-subscription planning. *Proceedings - International Conference on Automated Planning and Scheduling, ICAPS*, Whistler, BC, Canada, June 2004, 393-401.

22. Tang, H. and E. Miller-Hooks (2005). Algorithms for a stochastic selective travelling salesperson problem. *Journal of the Operational Research Society*, **56**, 439452.

23. Van Slyke, R.M. and R.J.-B. Wets (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, **17**, 638-663.

24. Wertz, J. and W. Larson (1999). *Space Mission Analysis and Design*. Microcosm Press, Torrance, CA.

25. Wolsey, L.A. (1998). *Integer Programming*. John Wiley & Sons, Inc., New York.

Appendix A. Single-Launch Code

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% AUTHOR: Capt Ben Kallemyn
%         AFIT/ENS
%         March 2007
%
% This program sets up the problem data for the
%
% SINGLE-LAUNCH Model vs HEURISTIC
%
% payload prioritization problem and
% calls GAMS to solve the model.
%
% There are 10 available requirements (not all must be used).
%   5 are multi-source requirements,
%   5 are sole-source requirements.
% There is space for up to 10 payloads for each requirement.
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
format compact

%set the budget levels and probability masses
budgets = [1000; 1500; 2000];
qs = [0.185; 0.63; 0.185];

Values = [];

tic

for i = 1:100

    %randomly generate payload data
    [payloads,as,cs,rs,beta1s,beta2s,beta3s,beta4s,beta5s] = randomizer(18);

    %calculate the total cost, weight power and volume of payloads
    totcost = sum(cs);
    rng = totcost - 250;%budget range
    totwei = sum(as(:,1));
    totpow = sum(as(:,2));
    totvol = sum(as(:,3));
```

```
42
43      %generate the bus capacities
44      bws(1) = .25*totwei+(totwei-.25*totwei)*rand;
45      bws(2) = .25*totpow+(totpow-.25*totpow)*rand;
46      bws(3) = .25*totvol+(totvol-.25*totvol)*rand;
47
48      %budgets for large variance scenarios
49  %     budgets(1) = 250+(.1*totcost-250)*rand;
50  %     budgets(2) = .2*totcost+(.4*totcost-.2*totcost)*rand;
51  %     budgets(3) = .9*totcost+(totcost-.9*totcost)*rand;
52
53      %budgets for low variance scenarios
54  %     budgets(2) = 250+(totcost-250)*rand;
55  %     budgets(1) = .85*budgets(2);
56  %     budgets(3) = 1.15*budgets(2);
57
58      %compute budgets using betainv (1,1), p = .05, .5, .95 for Pearson Tukey
59      %Uniform
60      budgets(1) = rng*betinv(.05,1,1)+250;
61      budgets(2) = rng*betinv(.5,1,1)+250;
62      budgets(3) = rng*betinv(.95,1,1)+250;
63      [Value1] = solver(budgets,qs,bws,payloads,as,cs,rs,beta1s,beta2s,beta3s,beta4s,beta5s);
64
65      %compute budgets using betainv (1.5,3), p = .05, .5, .95 for Pearson Tukey
66      %Left-Skewed
67      budgets(1) = rng*betinv(.05,1.5,3)+250;
68      budgets(2) = rng*betinv(.5,1.5,3)+250;
69      budgets(3) = rng*betinv(.95,1.5,3)+250;
70      [Value2] = solver(budgets,qs,bws,payloads,as,cs,rs,beta1s,beta2s,beta3s,beta4s,beta5s);
71
72
73      %compute budgets using betainv (5,5), p = .05, .5, .95 for Pearson Tukey
74      %Bell-Shaped
75      budgets(1) = rng*betinv(.05,5,5)+250;
76      budgets(2) = rng*betinv(.5,5,5)+250;
77      budgets(3) = rng*betinv(.95,5,5)+250;
78      [Value3] = solver(budgets,qs,bws,payloads,as,cs,rs,beta1s,beta2s,beta3s,beta4s,beta5s);
79
80      disp(i)
81
82      %store all results in one matrix
83      Values = [Values; Value1 Value2 Value3];
84
```

```
85
86 end
87
88 toc
```

```
 1 function[payloads,as,cs,rs,beta1s,beta2s,beta3s,beta4s,beta5s]=randomizer(p)
 2
 3 %bounds for requirements
 4
 5 %6 payloads
 6 if p == 6
 7 % Req  1    2    3    4    5    6    7    8    9    10
 8 bnd = [2    2    0    0    0    2    0    0    0    0;%lb of # of pylds
 9           2    2    0    0    0    2    0    0    0    0;%ub of # of pylds
10           0    0    0    0    0    10   10   10   10   10;%lb of reward
11           50   50   50   50   50   50   50   50   50   50;%ub of reward
12       5   5    5    5    5    5    5    5    5    1;%lb of cost * 50
13           20   20   20   20   20   20   20   20   20   20;%ub of cost * 50
14           1    1    1    1    1    1    1    1    1    1;%lb of weight * 50
15           20   20   20   20   20   20   20   20   20   20;%ub of weight * 50
16           1    1    1    1    1    1    1    1    1    1;%lb of power * 50
17           20   20   20   20   20   20   20   20   20   20;%ub of power * 50
18           2    2    2    2    2    2    2    2    2    2;%lb of volume
19           15   15   15   15   15   15   15   15   15   15];%ub of volume
20
21 elseif p == 9
22 %12 payloads
23 % Req  1    2    3    4    5    6    7    8    9    10
24 bnd = [3    3    0    0    0    3    0    0    0    0;%lb of # of pylds
25           3    3    0    0    0    3    0    0    0    0;%ub of # of pylds
26           0    0    0    0    0    10   10   10   10   10;%lb of reward
27           50   50   50   50   50   50   50   50   50   50;%ub of reward
28       5   5    5    5    5    5    5    5    5    1;%lb of cost * 50
29           20   20   20   20   20   20   20   20   20   20;%ub of cost * 50
30           1    1    1    1    1    1    1    1    1    1;%lb of weight * 50
31           20   20   20   20   20   20   20   20   20   20;%ub of weight * 50
32           1    1    1    1    1    1    1    1    1    1;%lb of power * 50
33           20   20   20   20   20   20   20   20   20   20;%ub of power * 50
34           2    2    2    2    2    2    2    2    2    2;%lb of volume
35           15   15   15   15   15   15   15   15   15   15];%ub of volume
36
37 elseif p == 12
38 %12 payloads
39 % Req  1    2    3    4    5    6    7    8    9    10
```

```
bnd = [3    3    0    0    0    3    3    0    0    0;%lb of # of pylds
       3    3    0    0    0    3    3    0    0    0;%ub of # of pylds
       0    0    0    0    0    10   10   10   10   10;%lb of reward
       50   50   50   50   50   50   50   50   50   50;%ub of reward
       5    5    5    5    5    5    5    5    5    1;%lb of cost * 50
       20   20   20   20   20   20   20   20   20   20;%ub of cost * 50
       1    1    1    1    1    1    1    1    1    1;%lb of weight * 50
       20   20   20   20   20   20   20   20   20   20;%ub of weight * 50
       1    1    1    1    1    1    1    1    1    1;%lb of power * 50
       20   20   20   20   20   20   20   20   20   20;%ub of power * 50
       2    2    2    2    2    2    2    2    2    2;%lb of volume
       15   15   15   15   15   15   15   15   15   15];%ub of volume

elseif p == 18
%20 payloads
% Req  1    2    3    4    5    6    7    8    9    10
bnd = [3    3    3    3    0    3    3    0    0    0;%lb of # of pylds
       3    3    3    3    0    3    3    0    0    0;%ub of # of pylds
       0    0    0    0    0    0    0    0    0    0;%lb of reward
       50   50   50   50   50   50   50   50   50   50;%ub of reward
       5    5    5    5    5    5    5    5    5    1;%lb of cost * 50
       20   20   20   20   20   20   20   20   20   20;%ub of cost * 50
       1    1    1    1    1    1    1    1    1    1;%lb of weight * 50
       20   20   20   20   20   20   20   20   20   20;%ub of weight * 50
       1    1    1    1    1    1    1    1    1    1;%lb of power * 50
       20   20   20   20   20   20   20   20   20   20;%ub of power * 50
       2    2    2    2    2    2    2    2    2    2;%lb of volume
       15   15   15   15   15   15   15   15   15   15];%ub of volume
end

%count the number of available payloads
k = zeros(1,10);
for i = 1:10
    k(i) = floor(bnd(1,i) + (bnd(2,i) - bnd(1,i) + 1) * rand);
end
payloads = sum(k);

%initialize multi-source reward values
beta1s = zeros(10,2);
beta2s = zeros(10,2);
beta3s = zeros(10,2);
beta4s = zeros(10,2);
beta5s = zeros(10,2);
```

```
83
84 %randomly generate multi-source rewards (diminishing)
85 for i = 1:5
86     u = bnd(4,i);
87     switch i
88         case 1
89             for j = 1:k(i)
90                 beta1s(j,1) = bnd(3,i)+(u-bnd(3,i)+1)*rand;
91                 u = beta1s(j,1);
92             end
93         case 2
94             for j = 1:k(i)
95                 beta2s(j,1) = bnd(3,i)+(u-bnd(3,i)+1)*rand;
96                 u = beta2s(j,1);
97             end
98         case 3
99             for j = 1:k(i)
100                beta3s(j,1) = bnd(3,i)+(u-bnd(3,i)+1)*rand;
101                u = beta3s(j,1);
102            end
103        case 4
104            for j = 1:k(i)
105                beta4s(j,1) = bnd(3,i)+(u-bnd(3,i)+1)*rand;
106                u = beta4s(j,1);
107            end
108        case 5
109            for j = 1:k(i)
110                beta5s(j,1) = bnd(3,i)+(u-bnd(3,i)+1)*rand;
111                u = beta5s(j,1);
112            end
113    end
114 end
115
116 %initialize sole-source rewards values
117 rs = zeros(5,1);
118
119 %generate sole-source reward values
120 for i = 1:5
121     if k(i+5) > 0
122         rs(i) = bnd(3,i+5)+(bnd(4,i+5)-bnd(3,i+5)+1)*rand;
123     end
124 end
125
```

```
%initialize payload parameters and costs
as = zeros(100,3);
cs = zeros(100,1);

%generate payload parameters and costs
for i = 1:10
    for j = 1:k(i)
        p = (10 * (i - 1)) + j;
        as(p,1) = 50*(bnd(7,i)+(bnd(8,i)-bnd(7,i)+1)*rand);
        as(p,2) = 50*(bnd(9,i)+(bnd(10,i)-bnd(9,i)+1)*rand);
        as(p,3) = (bnd(11,i)+(bnd(12,i)-bnd(11,i)+1)*rand);
        cs(p) = 50*(bnd(5,i)+(bnd(6,i)-bnd(5,i)+1)*rand);
    end
end
```

```
function [Value] = solver(budgets,qs,bws,payloads,as,cs,rs,beta1s,beta2s,beta3s,beta4s,beta5s)

%K is the row matrix of payload indices
K = [];
for k = 1:100
    if cs(k) ~= 0
        K = [K k];
    end
end

%number of payloads and scenarios for this instance
[z payloads] = size(K);
[scenarios z] = size(qs);

%Call the callgams function which invokes GAMS/CPLEX to solve the problem
[X1,T1,t1,X3,T3,t3] = callgams(payloads,scenarios,budgets,qs,bws,...
    as,cs,rs,beta1s,beta2s,beta3s,beta4s,beta5s);

%We need to compare the reward from GAMS (T3) to the expression q1*lorew +
%q2*medrew + q3*hirew also store the payload lists lolist, medlist, hilist


%Useful values for heuristics
pay = sum(X1.val,2);
cost = sum(pay.*cs);
wei = sum(pay.*as(:,1));
pow = sum(pay.*as(:,2));
vol = sum(pay.*as(:,3));
```

```
lobud = budgets(1);
medbud = budgets(2);
hibud = budgets(3);
ebud = sum(budgets.*qs);
reward = T1.val;

%Start the timer for the heuristic
tstart = clock;

%Run the low heuristic for the low budget
[lolist,loreward,locost] = lowheur(payloads,pay,reward,cost,...
    lobud,beta1s,beta2s,beta3s,beta4s,beta5s,rs,cs);

%Run the low heuristic for the med budget if it is lower than the expected
%budget and the high heuristic if it is greater
if medbud < ebud
    [medlist,medreward,medcost] = lowheur(payloads,pay,reward,cost,...
        medbud,beta1s,beta2s,beta3s,beta4s,beta5s,rs,cs);
elseif medbud > ebud
    [medlist,medreward,medcost] = hiheur(payloads,pay,reward,cost,...
        medbud,as,bws,beta1s,beta2s,beta3s,beta4s,beta5s,rs,cs);
else
    medlist = pay;
    medreward = reward;
    medcost = cost;
end

%Run the high heuristic for the high budget
[hilist,hireward,hicost]=hiheur(payloads,pay,reward,cost,hibud,...
    as,bws,beta1s,beta2s,beta3s,beta4s,beta5s,rs,cs);

%Stop the timer and update the computation time for the PS
timed = clock;
time = etime(clock,tstart);
t1.val = t1.val + time;

%The list of payloads and reward for the model are
list3 = sum(X3.val,2);
reward3 = T3.val;

%The reward vector for the heuristic is
rewards1 = [loreward; medreward; hireward];

```

```
%The weighted reward for the PS is
reward1 = sum(qs.*rewards1);

%Generate an output table summarizing results
rdiff = (reward3 - reward1)/reward1;
tdiff = abs(t3.val - t1.val)/t1.val;
Value = [reward3 reward1 rdiff];
```

```
function [X1,T1,t1,X3,T3,t3] = callgams(payloads,scenarios,budgets,qs,...
    bws,as,cs,rs,beta1s,beta2s,beta3s,beta4s,beta5s)

%Generate the arrays for the labels to be passed to GAMS
pay = num2str(payloads);
K = {};
I = {};
for i = 1:100
    k = ['K' int2str(i)];
    K = [K k];
end
for i = 1:payloads
    p = ['I' int2str(i)];
    I = [I p];
end
W = {};
for i = 1:scenarios
    w = ['W' int2str(i)];
    W = [W w];
end
S = {'WEI','POW','VOL'};
J = {'J6','J7','J8','J9','J10'};
P = {};
for i = 1:10
    p = ['P' int2str(i)];
    P = [P p];
end

%B3 is the budget levels for the PPR problem
B3.name = 'B';
B3.val = budgets;
B3.labels = {W};

%Q is the probability masses
Q.name = 'Q';
Q.val = qs;
```

```
Q.labels = {W};

%Calculate the expected budget
ebud = sum(budgets.*qs);

%B is the expected budget for the PS problem
B1.name = 'B';
B1.val = [ebud];

%BW is the engineering specification capacities for the satellite bus
BW.name = 'BW';
BW.val = bws;
BW.labels = {S};

%C is the costs for each payload
C.name = 'C';
C.val = cs;
C.labels = {K};

%A is the resource consumption for each payload
A.name = 'A';
A.val = as;
A.labels = {K,S};

%R is the sole-source reward
R.name = 'R';
R.val = rs;
R.labels = {J};

%BETA1 is reward for multi-source reward 1
BETA1.name = 'BETA1';
BETA1.val = beta1s;
BETA1.labels = {P,{'J1','J0'}};

%BETA2 is reward for multi-source reward 2
BETA2.name = 'BETA2';
BETA2.val = beta2s;
BETA2.labels = {P,{'J2','J0'}};

%BETA3 is reward for multi-source reward 3
BETA3.name = 'BETA3';
BETA3.val = beta3s;
BETA3.labels = {P,{'J3','J0'}};
```

```
80
81 %BETA4 is reward for multi-source reward 4
82 BETA4.name = 'BETA4';
83 BETA4.val = beta4s;
84 BETA4.labels = {P,{'J4','J0'}};
85
86 %BETA5 is reward for multi-source reward 5
87 BETA5.name = 'BETA5';
88 BETA5.val = beta5s;
89 BETA5.labels = {P,{'J5','J0'}};
90
91 %Call GAMS to solve the PPR(3) and PS(1) problems
92 %X is the solution for the decision variable Xkiw
93 %T is the maximum reward
94 %t is the computational time to solve the problem
95 [X3,T3,t3]=gams('J100','pay',B3,Q,BW,C,A,R,BETA1,BETA2,BETA3,BETA4,BETA5);
96 [X1,T1,t1]=gams('J100_1b','pay',B1,BW,C,A,R,BETA1,BETA2,BETA3,BETA4,BETA5);
```

```
1 ************************************************************************
2 *
3 * AUTHOR: Capt Ben Kallemyn
4 *         AFIT/ENS
5 *         March 2007
6 *
7 * This program solves the payload prioritization problem with requirements
8 *
9 * The following assumptions/limitations hold for this program:
10 *        1. There is a maximum of 10 requirements evenly split between
11 *           sole-source and multi-source types.
12 *        2. Associated with each requirement is space for 10 payloads.
13 *           If there is a need for fewer, fill in the spaces with zeros.
14 *        3. The reward functions for the multi-source requirements are
15 *           concave, piecewise linear. The slopes must be non-increasing.
16 *
17 ************************************************************************
18
19 *This sets the solver to CPLEX
20 OPTIONS MIP=CPLEX;
21
22 *This sets the global variable for the # of payloads
23 $setglobal pay 6
24 $if exist matglobs.gms $include matglobs.gms
25
26 *Define the sets
```

```
27 SETS
28 K available payload spaces / K1*K100 /
29 I priority levels / 1 * %pay% /
30 J requirements / J1*J10 /
31 W budget scenarios / W1*W3 /
32 SPEC engineering specifications / WEI, POW, VOL /
33 KK(K) payloads which are actually available
34 V(K) another name for the set of payloads
35 JM1(J) multi-source requirement 1
36 JM2(J) multi-source requirement 2
37 JM3(J) multi-source requirement 3
38 JM4(J) multi-source requirement 4
39 JM5(J) multi-source requirement 5
40 JS(J) sole-source requirements / J6*J10 /
41 JS6(JS) sole-source requirement 1
42 JS7(JS) sole-source requirement 2
43 JS8(JS) sole-source requirement 3
44 JS9(JS) sole-source requirement 4
45 JS10(JS) sole-source requirement 5
46 KJ(K) the set of payloads which satisfy the sole-source reqs
47 KJ1(K) the set of payloads which satisfy req 1
48 KJ2(K) the set of payloads which satisfy req 2
49 KJ3(K) the set of payloads which satisfy req 3
50 KJ4(K) the set of payloads which satisfy req 4
51 KJ5(K) the set of payloads which satisfy req 5
52 KJ6(K) the set of payloads which satisfy req 6
53 KJ7(K) the set of payloads which satisfy req 7
54 KJ8(K) the set of payloads which satisfy req 8
55 KJ9(K) the set of payloads which satisfy req 9
56 KJ10(K) the set of payloads which satisfy req 10
57 P counter for peicewise linear functions / P1*P10 / ;
58
59 PARAMETERS
60 B(W) budget scenarios
61        / W1  1500
62          W2  2000
63          W3  6000  /
64 BW(SPEC) specifications
65        / WEI  1900
66          POW  2600
67          VOL  22 /
68 C(K) cost of each payload
69        / K1       400
```

```
70          K2       200
71          K11      600
72          K12      500
73          K51      800
74          K52      950 /
75 Q(W) chance of budget scenario w
76        / W1  .2
77          W2  .6
78          W3  .2  /
79 R(JS) reward data
80        / J6   14
81          J7    0
82          J8    0
83          J9    0
84          J10   0 / ;
85
86 *Define set KK - the set of available payloads
87 LOOP(K$(C(K) NE 0), KK(K) = YES; V(K) = YES);
88
89 *Define the Kj subsets - the subset of payloads which satisfy each req
90 LOOP(K$(C(K) NE 0 AND ORD(K) GE  1 AND ORD(K) LE  10), KJ1(K) = YES);
91 LOOP(K$(C(K) NE 0 AND ORD(K) GE 11 AND ORD(K) LE  20), KJ2(K) = YES);
92 LOOP(K$(C(K) NE 0 AND ORD(K) GE 21 AND ORD(K) LE  30), KJ3(K) = YES);
93 LOOP(K$(C(K) NE 0 AND ORD(K) GE 31 AND ORD(K) LE  40), KJ4(K) = YES);
94 LOOP(K$(C(K) NE 0 AND ORD(K) GE 41 AND ORD(K) LE  50), KJ5(K) = YES);
95 LOOP(K$(C(K) NE 0 AND ORD(K) GE 51 AND ORD(K) LE  60), KJ6(K) = YES);
96 LOOP(K$(C(K) NE 0 AND ORD(K) GE 61 AND ORD(K) LE  70), KJ7(K) = YES);
97 LOOP(K$(C(K) NE 0 AND ORD(K) GE 71 AND ORD(K) LE  80), KJ8(K) = YES);
98 LOOP(K$(C(K) NE 0 AND ORD(K) GE 81 AND ORD(K) LE  90), KJ9(K) = YES);
99 LOOP(K$(C(K) NE 0 AND ORD(K) GE 91 AND ORD(K) LE 100), KJ10(K)= YES);
100
101 LOOP(KJ1$(C(KJ1) NE 0), JM1("J1") = YES);
102 LOOP(KJ2$(C(KJ2) NE 0), JM2("J2") = YES);
103 LOOP(KJ3$(C(KJ3) NE 0), JM3("J3") = YES);
104 LOOP(KJ4$(C(KJ4) NE 0), JM4("J4") = YES);
105 LOOP(KJ5$(C(KJ5) NE 0), JM5("J5") = YES);
106 LOOP(KJ6$(C(KJ6) NE 0), JS6("J6") = YES);
107 LOOP(KJ7$(C(KJ7) NE 0), JS7("J7") = YES);
108 LOOP(KJ8$(C(KJ8) NE 0), JS8("J8") = YES);
109 LOOP(KJ9$(C(KJ9) NE 0), JS9("J9") = YES);
110 LOOP(KJ10$(C(KJ10) NE 0), JS10("J10") = YES);
111
112 *Define set KJ - the set of payloads satisfying sole-source reqs
```

```
LOOP(K$(C(K) NE 0 AND ORD(K) GE 51), KJ(K)= YES);

TABLE A(K,SPEC) payload data
                    WEI      POW      VOL
         K1         200      350       3
         K2         100      450       2
         K11        400      300       4
         K12        300      500       4
         K51        900      750       9
         K52        800      700       7  ;

TABLE BETA1(P,J) slopes for peicewise liner reward function for req 1
                   J1
         P1         8
         P2         2 ;

TABLE BETA2(P,J) slopes for peicewise liner reward function for req 2
                   J2
         P1        10
         P2         5 ;

TABLE BETA3(P,J) slopes for peicewise liner reward function for req 3
                   J3
         P1         0
         P2         0 ;

TABLE BETA4(P,J) slopes for peicewise liner reward function for req 4
                   J4
         P1         0
         P2         0 ;

TABLE BETA5(P,J) slopes for peicewise liner reward function for req 5
                   J5
         P1         0
         P2         0 ;

VARIABLES
X(K,I,W) if payload k has priority level i and i is funded under scenario w
Y(K,J,W) if payload k satisfies requiremnt j under scenario w
Z(J,P,W) number of payloads which satisfy multi-source req j
T total expected reward ;

BINARY VARIABLES X,Y,Z;
```

```
156
157
158 EQUATIONS
159 REWARD define objective function
160 BUDGET(W) limit budget
161 SPECS(SPEC,W) limit specifications
162 PAYPRI(I,W) each priority level gets only 1 payload
163 ONEPRI(K,W) each payload has at most 1 priority level
164 CONSPRI(K,I,W) consecutive priority levels must be on list
165 CONSBUD(K,I,W) consecutive budget scenarios mmust be on list
166 ONEREQ6(JS,W) requirements 6 filled at most once per scenario
167 ONEREQ7(JS,W) requirements 7 filled at most once per scenario
168 ONEREQ8(JS,W) requirements 8 filled at most once per scenario
169 ONEREQ9(JS,W) requirements 9 filled at most once per scenario
170 ONEREQ10(JS,W) requirements 10 filled at most once per scenario
171 ONESOLE6(W)
172 ONESOLE7(W)
173 ONESOLE8(W)
174 ONESOLE9(W)
175 ONESOLE10(W)
176 DEFZ1(J,W) define decision variable z for req 1
177 DEFZ2(J,W) define decision variable z for req 2
178 DEFZ3(J,W) define decision variable z for req 3
179 DEFZ4(J,W) define decision variable z for req 4
180 DEFZ5(J,W) define decision variable z for req 5
181 LINKXY6(K,J,W) link decision variable y to x for req 6
182 LINKXY7(K,J,W) link decision variable y to x for req 7
183 LINKXY8(K,J,W) link decision variable y to x for req 8
184 LINKXY9(K,J,W) link decision variable y to x for req 9
185 LINKXY10(K,J,W) link decision variable y to x for req 10 ;
186
187
188 REWARD.. T =E= SUM(W, Q(W)*(
189          SUM((JM1,P),BETA1(P,JM1)*Z(JM1,P,W)) +
190          SUM((JM2,P),BETA2(P,JM2)*Z(JM2,P,W)) +
191          SUM((JM3,P),BETA3(P,JM3)*Z(JM3,P,W)) +
192          SUM((JM4,P),BETA4(P,JM4)*Z(JM4,P,W)) +
193          SUM((JM5,P),BETA5(P,JM5)*Z(JM5,P,W)) +
194          SUM((JS6,KJ6),R(JS6)*Y(KJ6,JS6,W)) +
195          SUM((JS7,KJ7),R(JS7)*Y(KJ7,JS7,W)) +
196          SUM((JS8,KJ8),R(JS8)*Y(KJ8,JS8,W)) +
197          SUM((JS9,KJ9),R(JS9)*Y(KJ9,JS9,W)) +
198          SUM((JS10,KJ10),R(JS10)*Y(KJ10,JS10,W))  )) ;
```

```
199 BUDGET(W).. SUM((KK,I), C(KK)*X(KK,I,W)) =L= B(W) ;
200 SPECS(SPEC,W).. SUM((KK,I), A(KK,SPEC)*X(KK,I,W)) =L= BW(SPEC) ;
201 PAYPRI(I,W).. SUM(KK, X(KK,I,W)) =L= 1 ;
202 ONEPRI(KK,W).. SUM(I, X(KK,I,W)) =L= 1 ;
203 CONSPRI(KK,I,W).. X(KK,I,W) =L= SUM(V, X(V,I-1,W)) + 1$(ORD(I) EQ 1) ;
204 CONSBUD(KK,I,W).. X(KK,I,W-1) =L= X(KK,I,W) ;
205 ONEREQ6(JS6,W).. SUM(KJ6, Y(KJ6,JS6,W)) =L= 1 ;
206 ONEREQ7(JS7,W).. SUM(KJ7, Y(KJ7,JS7,W)) =L= 1 ;
207 ONEREQ8(JS8,W).. SUM(KJ8, Y(KJ8,JS8,W)) =L= 1 ;
208 ONEREQ9(JS9,W).. SUM(KJ9, Y(KJ9,JS9,W)) =L= 1 ;
209 ONEREQ10(JS10,W).. SUM(KJ10, Y(KJ10,JS10,W)) =L= 1 ;
210 ONESOLE6(W).. SUM((KJ6,I), X(KJ6,I,W)) =L= 1 ;
211 ONESOLE7(W).. SUM((KJ7,I), X(KJ7,I,W)) =L= 1 ;
212 ONESOLE8(W).. SUM((KJ8,I), X(KJ8,I,W)) =L= 1 ;
213 ONESOLE9(W).. SUM((KJ9,I), X(KJ9,I,W)) =L= 1 ;
214 ONESOLE10(W).. SUM((KJ10,I), X(KJ10,I,W)) =L= 1 ;
215 DEFZ1(JM1,W).. SUM(P,Z(JM1,P,W)) =E= SUM((KJ1,I), X(KJ1,I,W)) ;
216 DEFZ2(JM2,W).. SUM(P,Z(JM2,P,W)) =E= SUM((KJ2,I), X(KJ2,I,W)) ;
217 DEFZ3(JM3,W).. SUM(P,Z(JM3,P,W)) =E= SUM((KJ3,I), X(KJ3,I,W)) ;
218 DEFZ4(JM4,W).. SUM(P,Z(JM4,P,W)) =E= SUM((KJ4,I), X(KJ4,I,W)) ;
219 DEFZ5(JM5,W).. SUM(P,Z(JM5,P,W)) =E= SUM((KJ5,I), X(KJ5,I,W)) ;
220 LINKXY6(KJ6,JS6,W).. Y(KJ6,JS6,W) =L= SUM(I, X(KJ6,I,W)) ;
221 LINKXY7(KJ7,JS7,W).. Y(KJ7,JS7,W) =L= SUM(I, X(KJ7,I,W)) ;
222 LINKXY8(KJ8,JS8,W).. Y(KJ8,JS8,W) =L= SUM(I, X(KJ8,I,W)) ;
223 LINKXY9(KJ9,JS9,W).. Y(KJ9,JS9,W) =L= SUM(I, X(KJ9,I,W)) ;
224 LINKXY10(KJ10,JS10,W).. Y(KJ10,JS10,W) =L= SUM(I, X(KJ10,I,W)) ;
225
226 MODEL SATELLITE /ALL/ ;
227
228 SATELLITE.OPTCR = 0.01;
229
230 $if exist matdata.gms $include matdata.gms
231
232 SOLVE SATELLITE USING MIP MAXIMIZING T ;
233
234 SCALAR COMPTIME time in CPU seconds to solve the model using CPLEX ;
235 COMPTIME = SATELLITE.RESUSD;
236
237 $libinclude matout X.l K I W
238 $libinclude matout T.l
239 $libinclude matout COMPTIME
240
```

```
241 DISPLAY X.L;
242 DISPLAY COMPTIME;

  1 ***************************************************************************
  2 *
  3 * AUTHOR: Capt Ben Kallemyn
  4 *         AFIT/ENS
  5 *         March 2007
  6 *
  7 * This program solves the payload prioritization problem with requirements
  8 *
  9 * The following assumptions/limitations hold for this program:
 10 *        1. There is a maximum of 10 requirements evenly split between
 11 *           sole-source and multi-source types.
 12 *        2. Associated with each requirement is space for 10 payloads.
 13 *           If there is a need for fewer, fill in the spaces with zeros.
 14 *        3. The reward functions for the multi-source requirements are
 15 *           concave, piecewise linear. The slopes must be non-increasing.
 16 *
 17 ***************************************************************************
 18
 19 *This sets the solver to CPLEX
 20 OPTIONS MIP=CPLEX;
 21
 22 *This sets the global variable for the # of payloads
 23 $setglobal pay 6
 24 $if exist matglobs.gms $include matglobs.gms
 25
 26 *Define the sets
 27 SETS
 28 K available payload spaces / K1*K100 /
 29 I priority levels / 1 * %pay% /
 30 J requirements / J1*J10 /
 31 SPEC engineering specifications / WEI, POW, VOL /
 32 KK(K) payloads which are actually available
 33 V(K) another name for the set of payloads
 34 JM1(J) multi-source requirement 1
 35 JM2(J) multi-source requirement 2
 36 JM3(J) multi-source requirement 3
 37 JM4(J) multi-source requirement 4
 38 JM5(J) multi-source requirement 5
 39 JS(J) sole-source requirements / J6*J10 /
 40 JS6(JS) sole-source requirement 1
 41 JS7(JS) sole-source requirement 2
```

```
JS8(JS) sole-source requirement 3
JS9(JS) sole-source requirement 4
JS10(JS) sole-source requirement 5
KJ(K) the set of payloads which satisfy the sole-source reqs
KJ1(K) the set of payloads which satisfy req 1
KJ2(K) the set of payloads which satisfy req 2
KJ3(K) the set of payloads which satisfy req 3
KJ4(K) the set of payloads which satisfy req 4
KJ5(K) the set of payloads which satisfy req 5
KJ6(K) the set of payloads which satisfy req 6
KJ7(K) the set of payloads which satisfy req 7
KJ8(K) the set of payloads which satisfy req 8
KJ9(K) the set of payloads which satisfy req 9
KJ10(K) the set of payloads which satisfy req 10
P counter for peicewise linear functions / P1*P10 / ;

PARAMETERS
B budget scenarios
       / 6000 /
BW(SPEC) specifications
       / WEI  1900
         POW  2600
         VOL  22 /
C(K) cost of each payload
       / K1       400
         K2       200
         K11      600
         K12      500
         K51      800
         K52      950 /
R(JS) reward data
       / J6   14
         J7    0
         J8    0
         J9    0
         J10   0 / ;

*Define set KK - the set of available payloads
LOOP(K$(C(K) NE 0), KK(K) = YES; V(K) = YES);

*Define the Kj subsets - the subset of payloads which satisfy each req
LOOP(K$(C(K) NE 0 AND ORD(K) GE  1 AND ORD(K) LE  10), KJ1(K) = YES);
LOOP(K$(C(K) NE 0 AND ORD(K) GE 11 AND ORD(K) LE  20), KJ2(K) = YES);
```

```
LOOP(K$(C(K) NE 0 AND ORD(K) GE 21 AND ORD(K) LE  30), KJ3(K) = YES);
LOOP(K$(C(K) NE 0 AND ORD(K) GE 31 AND ORD(K) LE  40), KJ4(K) = YES);
LOOP(K$(C(K) NE 0 AND ORD(K) GE 41 AND ORD(K) LE  50), KJ5(K) = YES);
LOOP(K$(C(K) NE 0 AND ORD(K) GE 51 AND ORD(K) LE  60), KJ6(K) = YES);
LOOP(K$(C(K) NE 0 AND ORD(K) GE 61 AND ORD(K) LE  70), KJ7(K) = YES);
LOOP(K$(C(K) NE 0 AND ORD(K) GE 71 AND ORD(K) LE  80), KJ8(K) = YES);
LOOP(K$(C(K) NE 0 AND ORD(K) GE 81 AND ORD(K) LE  90), KJ9(K) = YES);
LOOP(K$(C(K) NE 0 AND ORD(K) GE 91 AND ORD(K) LE 100), KJ10(K)= YES);

LOOP(KJ1$(C(KJ1) NE 0), JM1("J1") = YES);
LOOP(KJ2$(C(KJ2) NE 0), JM2("J2") = YES);
LOOP(KJ3$(C(KJ3) NE 0), JM3("J3") = YES);
LOOP(KJ4$(C(KJ4) NE 0), JM4("J4") = YES);
LOOP(KJ5$(C(KJ5) NE 0), JM5("J5") = YES);
LOOP(KJ6$(C(KJ6) NE 0), JS6("J6") = YES);
LOOP(KJ7$(C(KJ7) NE 0), JS7("J7") = YES);
LOOP(KJ8$(C(KJ8) NE 0), JS8("J8") = YES);
LOOP(KJ9$(C(KJ9) NE 0), JS9("J9") = YES);
LOOP(KJ10$(C(KJ10) NE 0), JS10("J10") = YES);

*Define set KJ - the set of payloads satisfying sole-source reqs
LOOP(K$(C(K) NE 0 AND ORD(K) GE 51), KJ(K)= YES);

TABLE A(K,SPEC) payload data
                   WEI      POW      VOL
         K1        200      350       3
         K2        100      450       2
         K11       400      300       4
         K12       300      500       4
         K51       900      750       9
         K52       800      700       7  ;

TABLE BETA1(P,J) slopes for peicewise liner reward function for req 1
                  J1
         P1        8
         P2        2 ;

TABLE BETA2(P,J) slopes for peicewise liner reward function for req 2
                  J2
         P1       10
         P2        5 ;

TABLE BETA3(P,J) slopes for peicewise liner reward function for req 3
```

```
                  J3
         P1        0
         P2        0  ;

TABLE BETA4(P,J) slopes for peicewise liner reward function for req 4
                  J4
         P1        0
         P2        0 ;

TABLE BETA5(P,J) slopes for peicewise liner reward function for req 5
                  J5
         P1        0
         P2        0 ;

VARIABLES
X(K,I) if payload k has priority level i and i is funded under scenario w
Y(K,J) if payload k satisfies requiremnt j under scenario w
Z(J,P) number of payloads which satisfy multi-source req j
T total expected reward ;

BINARY VARIABLES X,Y,Z;
*1,Z2,Z3,Z4,Z5;

EQUATIONS
REWARD define objective function
BUDGET limit budget
SPECS(SPEC) limit specifications
PAYPRI(I) each priority level gets only 1 payload
ONEPRI(K) each payload has at most 1 priority level
CONSPRI(K,I) consecutive priority levels must be on list
ONEREQ6(JS) requirements 6 filled at most once per scenario
ONEREQ7(JS) requirements 7 filled at most once per scenario
ONEREQ8(JS) requirements 8 filled at most once per scenario
ONEREQ9(JS) requirements 9 filled at most once per scenario
ONEREQ10(JS) requirements 10 filled at most once per scenario
ONESOLE6
ONESOLE7
ONESOLE8
ONESOLE9
ONESOLE10
DEFZ1(J) define decision variable z for req 1
DEFZ2(J) define decision variable z for req 2
DEFZ3(J) define decision variable z for req 3
```

```
171 DEFZ4(J) define decision variable z for req 4
172 DEFZ5(J) define decision variable z for req 5
173 LINKXY6(K,J) link decision variable y to x for req 6
174 LINKXY7(K,J) link decision variable y to x for req 7
175 LINKXY8(K,J) link decision variable y to x for req 8
176 LINKXY9(K,J) link decision variable y to x for req 9
177 LINKXY10(K,J) link decision variable y to x for req 10 ;
178
179
180 REWARD.. T =E= SUM((JM1,P),BETA1(P,JM1)*Z(JM1,P)) +
181          SUM((JM2,P),BETA2(P,JM2)*Z(JM2,P)) +
182          SUM((JM3,P),BETA3(P,JM3)*Z(JM3,P)) +
183          SUM((JM4,P),BETA4(P,JM4)*Z(JM4,P)) +
184          SUM((JM5,P),BETA5(P,JM5)*Z(JM5,P))  +
185          SUM((JS6,KJ6),R(JS6)*Y(KJ6,JS6)) +
186          SUM((JS7,KJ7),R(JS7)*Y(KJ7,JS7)) +
187          SUM((JS8,KJ8),R(JS8)*Y(KJ8,JS8)) +
188          SUM((JS9,KJ9),R(JS9)*Y(KJ9,JS9)) +
189          SUM((JS10,KJ10),R(JS10)*Y(KJ10,JS10)) ;
190 BUDGET.. SUM((KK,I), C(KK)*X(KK,I)) =L= B ;
191 SPECS(SPEC).. SUM((KK,I), A(KK,SPEC)*X(KK,I)) =L= BW(SPEC) ;
192 PAYPRI(I).. SUM(KK, X(KK,I)) =L= 1 ;
193 ONEPRI(KK).. SUM(I, X(KK,I)) =L= 1 ;
194 CONSPRI(KK,I).. X(KK,I) =L= SUM(V, X(V,I-1)) + 1$(ORD(I) EQ 1) ;
195 ONEREQ6(JS6).. SUM(KJ6, Y(KJ6,JS6)) =L= 1 ;
196 ONEREQ7(JS7).. SUM(KJ7, Y(KJ7,JS7)) =L= 1 ;
197 ONEREQ8(JS8).. SUM(KJ8, Y(KJ8,JS8)) =L= 1 ;
198 ONEREQ9(JS9).. SUM(KJ9, Y(KJ9,JS9)) =L= 1 ;
199 ONEREQ10(JS10).. SUM(KJ10, Y(KJ10,JS10)) =L= 1 ;
200 ONESOLE6.. SUM((KJ6,I), X(KJ6,I)) =L= 1 ;
201 ONESOLE7.. SUM((KJ7,I), X(KJ7,I)) =L= 1 ;
202 ONESOLE8.. SUM((KJ8,I), X(KJ8,I)) =L= 1 ;
203 ONESOLE9.. SUM((KJ9,I), X(KJ9,I)) =L= 1 ;
204 ONESOLE10.. SUM((KJ10,I), X(KJ10,I)) =L= 1 ;
205 DEFZ1(JM1).. SUM(P,Z(JM1,P)) =E= SUM((KJ1,I), X(KJ1,I)) ;
206 DEFZ2(JM2).. SUM(P,Z(JM2,P)) =E= SUM((KJ2,I), X(KJ2,I)) ;
207 DEFZ3(JM3).. SUM(P,Z(JM3,P)) =E= SUM((KJ3,I), X(KJ3,I)) ;
208 DEFZ4(JM4).. SUM(P,Z(JM4,P)) =E= SUM((KJ4,I), X(KJ4,I)) ;
209 DEFZ5(JM5).. SUM(P,Z(JM5,P)) =E= SUM((KJ5,I), X(KJ5,I)) ;
210 LINKXY6(KJ6,JS6).. Y(KJ6,JS6) =L= SUM(I, X(KJ6,I)) ;
211 LINKXY7(KJ7,JS7).. Y(KJ7,JS7) =L= SUM(I, X(KJ7,I)) ;
212 LINKXY8(KJ8,JS8).. Y(KJ8,JS8) =L= SUM(I, X(KJ8,I)) ;
213 LINKXY9(KJ9,JS9).. Y(KJ9,JS9) =L= SUM(I, X(KJ9,I)) ;
```

```
LINKXY10(KJ10,JS10).. Y(KJ10,JS10) =L= SUM(I, X(KJ10,I)) ;

MODEL SATELLITE /ALL/ ;

SATELLITE.OPTCR = 0.01;

$if exist matdata.gms $include matdata.gms

SOLVE SATELLITE USING MIP MAXIMIZING T ;

SCALAR COMPTIME time in CPU seconds to solve the model using CPLEX ;
COMPTIME = SATELLITE.RESUSD;

$libinclude matout X.l K I
$libinclude matout T.l
$libinclude matout COMPTIME

DISPLAY X.L;
DISPLAY COMPTIME;
```

```
function [lolist,loreward,locost] = lowheur(payloads,lolist,...
    loreward,locost,lobud,beta1s,beta2s,beta3s,beta4s,beta5s,rs,cs)

logoal = locost-lobud;

%loop until the cost is lower than the desired budget
while lobud < locost
    flag = 0;
    %calculate the minimum reward for each payload
    minrew = inf.*ones(100,1);

    %multi-source requirement min rewards
    for i = 1:5 %cycle through sole-source reqs
        num = 0; %initiate the counter
        for j = (10*(i-1))+1 : (10*i)
            num = num + lolist(j); %count the number of included payloads
        end
        switch i
            case 1 %update the min rewards for req 1
                for j = 1:10
                    if lolist(j) == 1 %only update minrew if available
                        %update with pw-linear function vals
                        minrew(j) = beta1s(num,1);
                    end
```

```
25                end
26            case 2 %update the min rewards for req 2
27                for j = 11:20
28                    if lolist(j) == 1 %only update minrew if available
29                        %update with pw-linear function vals
30                        minrew(j) = beta2s(num,1);
31                    end
32                end
33            case 3 %update the min rewards for req 3
34                for j = 21:30
35                    if lolist(j) == 1 %only update minrew if available
36                        %update with pw-linear function vals
37                        minrew(j) = beta3s(num,1);
38                    end
39                end
40            case 4 %update the min rewards for req 4
41                for j = 31:40
42                    if lolist(j) == 1 %only update minrew if available
43                        %update with pw-linear function vals
44                        minrew(j) = beta4s(num,1);
45                    end
46                end
47            case 5 %update the min rewards for req 5
48                for j = 41:50
49                    if lolist(j) == 1 %only update minrew if available
50                        %update with pw-linear function vals
51                        minrew(j) = beta5s(num,1);
52                    end
53                end
54        end
55    end
56
57    %sole-source requirement min rewards
58    for i = 6:10 %cycle through the sole-source reqs
59        for j = (10*(i-1))+1 : (10*i) %cycle through payloads for the req
60            if lolist(j) == 1 %if payload is included update the min reward
61                minrew(j) = rs(i-5);
62            end
63        end
64    end
65
66    lo = min(minrew); %find the min of the min rewards
67
```

```
    %Generate lorew - the list of payload indices which have the minimal
    %reward
    lorew = [];
    for i = 1:100
        if minrew(i) == lo
            kp = i;
            lorew = [lorew kp]; %row vector of payload indices
        end
    end

    %Iterate through lolist to find largest payload cost
    mcost = 0;
    for i = lorew
        if cs(i) > mcost
            %track the least cost payload
            mcost = cs(i);
            maxcost = i;
        end
    end

    lolist(maxcost) = 0;
    loreward = loreward - lo;

    %Update the locost
    locost = sum(lolist.*cs);

end
```

```
function [hilist,hireward,hicost] = hiheur(payloads,hilist,hireward,...
    hicost,hibud,as,bws,beta1s,beta2s,beta3s,beta4s,beta5s,rs,cs)

%Generate list of available payloads
%Only multi-source payloads and sole-source reqs whose payloads haven't
%already been included
avail = zeros(100,1);

%if payload satisfies multi-source req add it to avail list
for i = 1:50
    if hilist(i) == 0 & cs(i) > 0 %check if payload exists & not included
        avail(i) = 1;
    end
end

%if payload satisfies sole-source req add it to avail list
```

```
for i = 6:10
    num = 0;
    cst = 0;
    %Check to see if payload exists and none are included for each req
    for j = (10*(i-1))+1 : (10*i)
        num = num + hilist(j); %the number of included payloads
        cst = cst + cs(j); % the cost of the included payloads
    end
    %if the req is not satisfied and there are payloads available
    if num ~= 1 & cst > 0
        for j = (10*(i-1))+1 : (10*i)
            if cs(j) > 0 %if the payload exists
                avail(j) = 1; %add the payload to the avail list
            end
        end
    end
end

%while there are still payloads available
while sum(avail) ~= 0

    %calculate the maximum reward for each payload
    maxrew = zeros(100,1);

    %multi-source requirement min rewards
    for i = 1:5 %cycle through sole-source reqs
        num = 0; %initiate the counter
        for j = (10*(i-1))+1 : (10*i)
            num = num + hilist(j); %count the number of included payloads
        end
        switch i
            case 1 %update the max rewards for req 1
                for j = 1:10
                    if avail(j) == 1 %only update maxrew if available
                        %update with pw-linear function vals
                        maxrew(j) = beta1s(num+1,1);
                    end
                end
            case 2 %update the max rewards for req 2
                for j = 11:20
                    if avail(j) == 1 %only update maxrew if available
                        %update with pw-linear function vals
                        maxrew(j) = beta2s(num+1,1);
```

```
60                  end
61              end
62          case 3 %update the max rewards for req 3
63              for j = 21:30
64                  if avail(j) == 1 %only update maxrew if available
65                      %update with pw-linear function vals
66                      maxrew(j) = beta3s(num+1,1);
67                  end
68              end
69          case 4 %update the max rewards for req 4
70              for j = 31:40
71                  if avail(j) == 1 %only update maxrew if available
72                      %update with pw-linear function vals
73                      maxrew(j) = beta4s(num+1,1);
74                  end
75              end
76          case 5 %update the max rewards for req 5
77              for j = 41:50
78                  if avail(j) == 1 %only update maxrew if available
79                      %update with pw-linear function vals
80                      maxrew(j) = beta5s(num+1,1);
81                  end
82              end
83      end
84  end
85
86  %sole-source requirement max rewards
87  for i = 6:10 %cycle through the sole-source reqs
88      for j = (10*(i-1))+1 : (10*i) %cycle through payloads for the req
89          if avail(j) == 1 %if payload is included update the max reward
90              maxrew(j) = rs(i-5);
91          end
92      end
93  end
94
95  hi = max(maxrew);
96
97  if hi == 0
98      avail = zeros(100,1);
99  else
100
101     %Generate hirew - the list of payload indices which have maximal reward
102     hirew = [];
```

```
103         for i = 1:100
104             if maxrew(i) == hi
105                 kp = i;
106                 hirew = [hirew kp];
107             end
108         end
109 
110         %Iterate through hilist to find least payload cost
111         mcost = inf;
112         for i = hirew
113             if cs(i) < mcost
114                 %track the least cost payload
115                 mcost = cs(i);
116                 mincost = i;
117             end
118         end
119 
120         %Check feasibility of budget, weight, power and volume
121         if hibud >= hicost + cs(mincost) &...
122                 bws(1) >= sum(hilist.*as(:,1)) + as(mincost,1) &...
123                 bws(2) >= sum(hilist.*as(:,2)) + as(mincost,2) &...
124                 bws(3) >= sum(hilist.*as(:,3)) + as(mincost,3)
125             %update the list and reward
126             hilist(mincost) = 1;
127             hicost = hicost + cs(mincost);
128             hireward = hireward + hi;
129             maxrew(mincost) = 0;
130             avail(mincost) = 0;
131             %remove all sole-source payloads from available list if one is added
132             if mincost >= 91
133                 for z = 91:100
134                     avail(z) = 0;
135                 end
136             elseif mincost >= 81
137                 for z = 81:90
138                     avail(z) = 0;
139                 end
140             elseif mincost >= 71
141                 for z = 71:80
142                     avail(z) = 0;
143                 end
144             elseif mincost >= 61
145                 for z = 61:70
```

```
146                     avail(z) = 0;
147                 end
148             elseif mincost >= 51
149                 for z = 51:60
150                     avail(z) = 0;
151                 end
152             end
153         else %just remove the payload from the available list
154             avail(mincost) = 0;
155         end
156     end
157 end
```

Appendix B. Multiple-Launch Code

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% AUTHOR: Capt Ben Kallemyn
%         AFIT/ENS
%         March 2007
%
% This program sets up the problem data for the
%
% MULTIPLE-LAUNCH
%
% payload prioritization problem and
% calls GAMS to solve the model.
%
% There are 10 available requirements (not all must be used).
%   5 are multi-source requirements,
%   5 are sole-source requirements.
% There is space for up to 10 payloads for each requirement.
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
format compact

%Enter the number of payloads, scenarios and launches for this instance
payloads = 12;
scenarios = 3;
launches = 8;
life=3;
pws = 3;
qs = [.185 .63 .185];

Values = [];

for n = 1:100
    n

    %initialize reward values
    beta1sL = zeros(2,2,launches);
    beta2sL = zeros(2,2,launches);
    beta3sL = zeros(2,2,launches);
    beta4sL = zeros(2,2,launches);
```

```
42      beta5sL = zeros(2,2,launches);
43      rsL = zeros(5,launches);
44
45      %generate the payload and reward data
46      [payloads,as,cs,rsL,beta1sL,beta2sL,beta3sL,...
47          beta4sL,beta5sL] = multrand(12,launches,life);
48
49      %check the size of the reward matrices - if the rewards are empty, fill with zeros
50      [m1 n] = size(beta1sL);
51      [m2 n] = size(beta2sL);
52      [m3 n] = size(beta3sL);
53      [m4 n] = size(beta4sL);
54      [m5 n] = size(beta5sL);
55      if m2 == 0
56          beta2sL=zeros(m1,2,launches);
57      end
58      if m3 == 0
59          beta3sL=zeros(m1,2,launches);
60      end
61      if m4 == 0
62          beta4sL=zeros(m1,2,launches);
63      end
64      if m5 == 0
65          beta5sL=zeros(m1,2,launches);
66      end
67
68      %calculate the total cost, weight power and volume of payloads
69      totcost = sum(cs);
70      totwei = sum(as(:,1));
71      totpow = sum(as(:,2));
72      totvol = sum(as(:,3));
73
74      %initialize budget ranges
75      rng = zeros(1,launches);
76
77      for l = 1:launches
78          %budgets for large variance scenarios
79  %         budgets(l,1) = 50*(5+(.1*totcost(l)-5)*rand);
80  %         budgets(l,2) = 50*(.2*totcost(l)+(.4*totcost(l)-.2*totcost(l))*rand);
81  %         budgets(l,3) = 50*(.9*totcost(l)+(totcost(l)-.9*totcost(l))*rand);
82
83          rng(l) = totcost(l) - 250;%budget range
84
```

```
        %compute budgets using betainv (1.5,3), p = .05, .5, .95 for Pearson Tukey
        %Left-Skewed
        budgets(l,1) = rng(l)*betinv(.05,1.5,3)+250;
        budgets(l,2) = rng(l)*betinv(.5,1.5,3)+250;
        budgets(l,3) = rng(l)*betinv(.95,1.5,3)+250;

        %generate the bus capacities
        bws(l,1) = (.25*totwei+(totwei-.25*totwei)*rand);
        bws(l,2) = (.25*totpow+(totpow-.25*totpow)*rand);
        bws(l,3) = (.25*totvol+(totvol-.25*totvol)*rand);
    end

    %Solve the problem for three lives
    life = 3;
    [Xf,Tf,tf,Xs,Ts,ts]=gamscall(payloads,scenarios,launches,life,budgets,...
        pws,qs,bws,as,cs,rsL,beta1sL,beta2sL,beta3sL,beta4sL,beta5sL);
    imp = 100*(Tf.val/Ts.val - 1);
    Value1 = [Tf.val Ts.val imp];

    life = 4;
    [Xf,Tf,tf,Xs,Ts,ts]=gamscall(payloads,scenarios,launches,life,budgets,...
        pws,qs,bws,as,cs,rsL,beta1sL,beta2sL,beta3sL,beta4sL,beta5sL);
    imp = 100*(Tf.val/Ts.val - 1);
    Value2 = [Tf.val Ts.val imp];

    life = 5;
    [Xf,Tf,tf,Xs,Ts,ts]=gamscall(payloads,scenarios,launches,life,budgets,...
        pws,qs,bws,as,cs,rsL,beta1sL,beta2sL,beta3sL,beta4sL,beta5sL);
    imp = 100*(Tf.val/Ts.val - 1);
    Value3 = [Tf.val Ts.val imp];

    %store all results in one matrix
    Values = [Values; Value1 Value2 Value3];
end
```

```
function[payloads,as,cs,rsL,beta1sL,beta2sL,beta3sL,...
    beta4sL,beta5sL] = multrand(p,launches,life)

%bounds for requirements

%6 payloads
if p == 6
% Req  1    2    3    4    5    6    7    8    9    10
bnd = [2    2    0    0    0    2    0    0    0    0;%lb of # of pylds
```

```
            2    2    0    0    0    2    0    0    0    0;%ub of # of pylds
            0    0    0    0    0    10   10   10   10   10;%lb of reward
            50   50   50   50   50   50   50   50   50   50;%ub of reward
        5    5    5    5    5    5    5    5    5    1;%lb of cost * 50
            20   20   20   20   20   20   20   20   20   20;%ub of cost * 50
            1    1    1    1    1    1    1    1    1    1;%lb of weight * 50
            20   20   20   20   20   20   20   20   20   20;%ub of weight * 50
            1    1    1    1    1    1    1    1    1    1;%lb of power * 50
            20   20   20   20   20   20   20   20   20   20;%ub of power * 50
            2    2    2    2    2    2    2    2    2    2;%lb of volume
            15   15   15   15   15   15   15   15   15   15];%ub of volume

elseif p == 12
%12 payloads
% Req  1    2    3    4    5    6    7    8    9    10
bnd = [3    3    0    0    0    3    3    0    0    0;%lb of # of pylds
            3    3    0    0    0    3    3    0    0    0;%ub of # of pylds
            0    0    0    0    0    10   10   10   10   10;%lb of reward
            50   50   50   50   50   50   50   50   50   50;%ub of reward
        5    5    5    5    5    5    5    5    5    1;%lb of cost * 50
            20   20   20   20   20   20   20   20   20   20;%ub of cost * 50
            1    1    1    1    1    1    1    1    1    1;%lb of weight * 50
            20   20   20   20   20   20   20   20   20   20;%ub of weight * 50
            1    1    1    1    1    1    1    1    1    1;%lb of power * 50
            20   20   20   20   20   20   20   20   20   20;%ub of power * 50
            2    2    2    2    2    2    2    2    2    2;%lb of volume
            15   15   15   15   15   15   15   15   15   15];%ub of volume

elseif p == 18
%20 payloads
% Req  1    2    3    4    5    6    7    8    9    10
bnd = [3    3    3    3    0    3    3    0    0    0;%lb of # of pylds
            3    3    3    3    0    3    3    0    0    0;%ub of # of pylds
            0    0    0    0    0    0    0    0    0    0;%lb of reward
            50   50   50   50   50   50   50   50   50   50;%ub of reward
        5    5    5    5    5    5    5    5    5    1;%lb of cost * 50
            20   20   20   20   20   20   20   20   20   20;%ub of cost * 50
            1    1    1    1    1    1    1    1    1    1;%lb of weight * 50
            20   20   20   20   20   20   20   20   20   20;%ub of weight * 50
            1    1    1    1    1    1    1    1    1    1;%lb of power * 50
            20   20   20   20   20   20   20   20   20   20;%ub of power * 50
            2    2    2    2    2    2    2    2    2    2;%lb of volume
            15   15   15   15   15   15   15   15   15   15];%ub of volume
```

```
53 end
54
55 %count the number of available payloads
56 k = zeros(1,10);
57 for i = 1:10
58     k(i) = floor(bnd(1,i) + (bnd(2,i) - bnd(1,i) + 1) * rand);
59 end
60 payloads = sum(k);
61
62 %initialize reward values
63 beta1sL = zeros(k(1)*life,2,launches);
64 beta2sL = zeros(k(2)*life,2,launches);
65 beta3sL = zeros(k(3)*life,2,launches);
66 beta4sL = zeros(k(4)*life,2,launches);
67 beta5sL = zeros(k(5)*life,2,launches);
68 rsL = zeros(5,launches);
69
70 %randomly generate multi-source rewards (diminishing)
71 for l = 1:launches
72     for i = 1:5
73         u = bnd(4,i);
74         switch i
75             case 1
76                 for j = 1:(k(i)*life)
77                     beta1sL(j,1,l) = (bnd(3,i)+(u-bnd(3,i)+1)*rand);
78                     u = beta1sL(j,1,l);
79                 end
80             case 2
81                 for j = 1:(k(i)*life)
82                     beta2sL(j,1,l) = (bnd(3,i)+(u-bnd(3,i)+1)*rand);
83                     u = beta2sL(j,1,l);
84                 end
85             case 3
86                 for j = 1:(k(i)*life)
87                     beta3sL(j,1,l) = (bnd(3,i)+(u-bnd(3,i)+1)*rand);
88                     u = beta3sL(j,1,l);
89                 end
90             case 4
91                 for j = 1:(k(i)*life)
92                     beta4sL(j,1,l) = (bnd(3,i)+(u-bnd(3,i)+1)*rand);
93                     u = beta4sL(j,1,l);
94                 end
95             case 5
```

```
                for j = 1:(k(i)*life)
                    beta5sL(j,1,l) = (bnd(3,i)+(u-bnd(3,i)+1)*rand);
                    u = beta5sL(j,1,l);
                end
        end
    end

    %generate sole-source reward values
    for i = 1:5
        if k(i+5) > 0
            rsL(i,l) = (bnd(3,i+5)+(bnd(4,i+5)-bnd(3,i+5)+1)*rand);
        end
    end
end

%initialize payload parameters and costs
as = zeros(100,3);
cs = zeros(100,launches);

%generate payload parameters and costs
for i = 1:10
    for j = 1:k(i)
        p = (10 * (i - 1)) + j;
        as(p,1) = 50*(bnd(7,i)+(bnd(8,i)-bnd(7,i)+1)*rand);
        as(p,2) = 50*(bnd(9,i)+(bnd(10,i)-bnd(9,i)+1)*rand);
        as(p,3) = (bnd(11,i)+(bnd(12,i)-bnd(11,i)+1)*rand);
        for l = 1:launches
            cs(p,l) = 50*(bnd(5,i)+(bnd(6,i)-bnd(5,i)+1)*rand);
        end
    end
end
```

```
function[Xf,Tf,tf,Xs,Ts,ts] = gamscall(payloads,scenarios,launches,life,...
    budgets,pws,qs,bws,as,cs,rsL,beta1sL,beta2sL,beta3sL,beta4sL,beta5sL)

%Generate the arrays for the labels to be passed to GAMS
pay = num2str(payloads);
launch = num2str(launches);
[m n] = size(beta1sL);

K = {};
for i = 1:100
    k = ['K' int2str(i)];
```

```
    K = [K k];
end
W = {};
for i = 1:scenarios
    w = ['W' int2str(i)];
    W = [W w];
end
L = {};
for i = 1:launches
    l = [int2str(i)];
    L = [L l];
end
S = {'WEI','POW','VOL'};
J = {'J6','J7','J8','J9','J10'};
P = {};
for i = 1:m
    p = [int2str(i)];
    P = [P p];
end

%B3 is the budget levels
B3.name = 'B';
B3.val = budgets;
B3.labels = {L,W};

%Q is the probability masses
Q.name = 'Q';
Q.val = qs;
Q.labels = {W};

%BW is the engineering specification capacities for the satellite bus
BW.name = 'BW';
BW.val = bws;
BW.labels = {L,S};

%C is the costs for each payload
C.name = 'C';
C.val = cs;
C.labels = {K,L};

%A is the resource consumption for each payload
A.name = 'A';
A.val = as;
```

```
A.labels = {K,S};

%R is the sole-source reward
R.name = 'R';
R.val = rsL;
R.labels = {J,L};

%BETA1 is reward for multi-source reward 1
BETA1.name = 'BETA1';
BETA1.val = beta1sL;
BETA1.labels = {P,{'J1','J0'},L};

%BETA2 is reward for multi-source reward 2
BETA2.name = 'BETA2';
BETA2.val = beta2sL;
BETA2.labels = {P,{'J2','J0'},L};

%BETA3 is reward for multi-source reward 3
BETA3.name = 'BETA3';
BETA3.val = beta3sL;
BETA3.labels = {P,{'J3','J0'},L};

%BETA4 is reward for multi-source reward 4
BETA4.name = 'BETA4';
BETA4.val = beta4sL;
BETA4.labels = {P,{'J4','J0'},L};

%BETA5 is reward for multi-source reward 5
BETA5.name = 'BETA5';
BETA5.val = beta5sL;
BETA5.labels = {P,{'J5','J0'},L};

%Call GAMS to solve the PPR(3) and PS(1) problems
%X is the solution for the decision variable Xkiw
%T is the maximum reward
%t is the computational time to solve the problem
pw = num2str(pws*life);
life = num2str(life);
[Xf,Tf,tf]= gams('FMLP','pay','launch','life','pw',B3,Q,BW,C,A,R,BETA1,BETA2,BETA3,BETA4,BETA5);
[Xs,Ts,ts]= gams('SMLP','pay','launch','life','pw',B3,Q,BW,C,A,R,BETA1,BETA2,BETA3,BETA4,BETA5);
```

```
************************************************************************
*
* AUTHOR: Capt Ben Kallemyn
```

```
4 *         AFIT/ENS
5 *         March 2007
6 *
7 * This program solves the multiple-launch memoryless prioritization problem
8 *
9 * The following assumptions/limitations hold for this program:
10 *        1. There is a maximum of 10 requirements evenly split between
11 *           sole-source and multi-source types.
12 *        2. Associated with each requirement is space for 10 payloads.
13 *           If there is a need for fewer, fill in the spaces with zeros.
14 *        3. The reward functions for the multi-source requirements are
15 *           concave, piecewise linear. The slopes must be non-increasing.
16 *        4. If a payload is available, it is available for all launches.
17 *           There should be a cost for each launch for each included payload.
18 *
19 ************************************************************************
20
21 *This sets the solver to CPLEX
22 OPTIONS MIP=CPLEX;
23
24 *This sets the global variable for the # of payloads
25 $setglobal pay 6
26 $setglobal launch 2
27 $setglobal life 2
28 $setglobal pw 4
29 *pw is the number of reqs that can be in const (# payloads per req * life)
30 $if exist matglobs.gms $include matglobs.gms
31
32 *Define the sets
33 SETS
34 K available payload spaces / K1*K100 /
35 I priority levels / 1 * %pay% /
36 J requirements / J1*J10 /
37 L launches / 1 * %launch% /
38 W budget scenarios / W1*W3 /
39 SPEC engineering specifications / WEI, POW, VOL /
40 KK(K) payloads which are actually available
41 V(K) another name for the set of payloads
42 JM1(J) multi-source requirement 1
43 JM2(J) multi-source requirement 2
44 JM3(J) multi-source requirement 3
45 JM4(J) multi-source requirement 4
46 JM5(J) multi-source requirement 5
```

```
47 JS(J) sole-source requirements / J6*J10 /
48 JS6(JS) sole-source requirement 1
49 JS7(JS) sole-source requirement 2
50 JS8(JS) sole-source requirement 3
51 JS9(JS) sole-source requirement 4
52 JS10(JS) sole-source requirement 5
53 KJ(K) the set of payloads which satisfy the sole-source reqs
54 KJ1(K) the set of payloads which satisfy req 1
55 KJ2(K) the set of payloads which satisfy req 2
56 KJ3(K) the set of payloads which satisfy req 3
57 KJ4(K) the set of payloads which satisfy req 4
58 KJ5(K) the set of payloads which satisfy req 5
59 KJ6(K) the set of payloads which satisfy req 6
60 KJ7(K) the set of payloads which satisfy req 7
61 KJ8(K) the set of payloads which satisfy req 8
62 KJ9(K) the set of payloads which satisfy req 9
63 KJ10(K) the set of payloads which satisfy req 10
64 P counter for peicewise linear functions / 1 * %pw% /
65 NK(K);
66
67 ALIAS (L,LW) ;
68 V(K) = YES;
69
70 PARAMETERS
71 Q(W) chance of budget scenario w
72        / W1  .2
73          W2  .6
74          W3  .2  / ;
75
76 TABLE C(K,L)
77                      1        2
78          K1          400      400
79          K2          200      300
80          K11         600      700
81          K12         500      600
82          K51         800      800
83          K52         950      900 ;
84
85 *Define set KK - the set of available payloads
86 LOOP(K$(C(K,'1') NE 0), KK(K) = YES; V(K) = YES);
87 LOOP(K$(C(K,'1') EQ 0), NK(K) = YES);
88
89 *Define the Kj subsets - the subset of payloads which satisfy each req
```

```
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE  1 AND ORD(K) LE  10), KJ1(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 11 AND ORD(K) LE  20), KJ2(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 21 AND ORD(K) LE  30), KJ3(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 31 AND ORD(K) LE  40), KJ4(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 41 AND ORD(K) LE  50), KJ5(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 51 AND ORD(K) LE  60), KJ6(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 61 AND ORD(K) LE  70), KJ7(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 71 AND ORD(K) LE  80), KJ8(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 81 AND ORD(K) LE  90), KJ9(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 91 AND ORD(K) LE 100), KJ10(K)= YES);

LOOP(KJ1$(C(KJ1,'1') NE 0), JM1("J1") = YES);
LOOP(KJ2$(C(KJ2,'1') NE 0), JM2("J2") = YES);
LOOP(KJ3$(C(KJ3,'1') NE 0), JM3("J3") = YES);
LOOP(KJ4$(C(KJ4,'1') NE 0), JM4("J4") = YES);
LOOP(KJ5$(C(KJ5,'1') NE 0), JM5("J5") = YES);
LOOP(KJ6$(C(KJ6,'1') NE 0), JS6("J6") = YES);
LOOP(KJ7$(C(KJ7,'1') NE 0), JS7("J7") = YES);
LOOP(KJ8$(C(KJ8,'1') NE 0), JS8("J8") = YES);
LOOP(KJ9$(C(KJ9,'1') NE 0), JS9("J9") = YES);
LOOP(KJ10$(C(KJ10,'1') NE 0), JS10("J10") = YES);

*Define set KJ - the set of payloads satisfying sole-source reqs
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 51), KJ(K)= YES);

TABLE A(K,SPEC) payload data
                    WEI      POW      VOL
         K1         200      350       3
         K2         100      450       2
         K11        400      300       4
         K12        300      500       4
         K51        900      750       9
         K52        800      700       7  ;

TABLE R(JS,L)
             1  2
         J6  14 21
         J7  0  0
         J8  0  0
         J9  0  0
         J10 0  0  ;
```

```
133
134 TABLE BW(L,SPEC)
135              WEI  POW  VOL
136           1  1900 2600 22
137           2  1500 2300 18 ;
138
139 TABLE B(L,W)
140              W1   W2   W3
141           1  1500 2000 2500
142           2  1200 1600 2000 ;
143
144 TABLE BETA1(P,J,L)
145                  J1.1     J1.2
146          1        8        10
147          2        2        5
148          3        0        0
149          4        0        0 ;
150
151 TABLE BETA2(P,J,L)
152                  J2.1     J2.2
153          1        9        5
154          2        3        5
155          3        1        3
156          4        0        1 ;
157
158 TABLE BETA3(P,J,L)
159                  J3.1     J3.2
160          1        0        0
161          2        0        0 ;
162
163 TABLE BETA4(P,J,L)
164                  J4.1     J4.2
165          1        0        0
166          2        0        0  ;
167
168 TABLE BETA5(P,J,L)
169                  J5.1     J5.2
170          1        0        0
171          2        0        0  ;
172
173 VARIABLES
174 X(K,I,L,W) if payload k has priority level i and i is funded under scenario w
175 Y(K,J,L,W) if payload k satisfies requiremnt j under scenario w
```

```
176 IY(J,L,W) if sole-source req j is satisfied in const at launch l
177 Z(J,L,W) number of payloads which satisfy multi-source req j on launch l
178 IZ(J,P,L,W) number of payloads which satisfy multi-source req j in const at l
179 T total expected reward ;
180
181 BINARY VARIABLES X,Y,IY,IZ;
182
183 EQUATIONS
184 REWARD define objective function
185 BUDGET(L,W) limit budget
186 SPECS(SPEC,L,W) limit specifications
187 PAYPRI(I,L,W) each priority level gets only 1 payload
188 ONEPRI(K,L,W) each payload has at most 1 priority level
189 CONSPRI(K,I,L,W) consecutive priority levels must be on list
190 CONSBUD(K,I,L,W) consecutive budget scenarios mmust be on list
191 ONEREQ6(JS,L,W)
192 ONEREQ7(JS,L,W)
193 ONEREQ8(JS,L,W)
194 ONEREQ9(JS,L,W)
195 ONEREQ10(JS,L,W)
196 ONESOLE6(L,W)
197 ONESOLE7(L,W)
198 ONESOLE8(L,W)
199 ONESOLE9(L,W)
200 ONESOLE10(L,W)
201 DEFZ1(J,L,W) define decision variable z for req 1
202 DEFZ2(J,L,W) define decision variable z for req 2
203 DEFZ3(J,L,W) define decision variable z for req 3
204 DEFZ4(J,L,W) define decision variable z for req 4
205 DEFZ5(J,L,W) define decision variable z for req 5
206 LINKXY6(K,J,L,W) link decision variable y to x for req 6
207 LINKXY7(K,J,L,W) link decision variable y to x for req 7
208 LINKXY8(K,J,L,W) link decision variable y to x for req 8
209 LINKXY9(K,J,L,W) link decision variable y to x for req 9
210 LINKXY10(K,J,L,W) link decision variable y to x for req 10
211 DEFIZ1(J,L,W)
212 DEFIZ2(J,L,W)
213 DEFIZ3(J,L,W)
214 DEFIZ4(J,L,W)
215 DEFIZ5(J,L,W)
216 DEFIY6(J,L,W)
217 DEFIY7(J,L,W)
218 DEFIY8(J,L,W)
```

```
DEFIY9(J,L,W)
DEFIY10(J,L,W)
XNK(K,I,L,W) ;


REWARD.. T =E= SUM(W, Q(W)*SUM(L,(
         SUM((JM1,P),BETA1(P,JM1,L)*IZ(JM1,P,L,W)) +
         SUM((JM2,P),BETA2(P,JM2,L)*IZ(JM2,P,L,W)) +
         SUM((JM3,P),BETA3(P,JM3,L)*IZ(JM3,P,L,W)) +
         SUM((JM4,P),BETA4(P,JM4,L)*IZ(JM4,P,L,W)) +
         SUM((JM5,P),BETA5(P,JM5,L)*IZ(JM5,P,L,W)) +
         SUM(JS6,R(JS6,L)*IY(JS6,L,W)) +
         SUM(JS7,R(JS7,L)*IY(JS7,L,W)) +
         SUM(JS8,R(JS8,L)*IY(JS8,L,W)) +
         SUM(JS9,R(JS9,L)*IY(JS9,L,W)) +
         SUM(JS10,R(JS10,L)*IY(JS10,L,W))  ))) ;
BUDGET(L,W).. SUM((KK,I), C(KK,L)*X(KK,I,L,W)) =L= B(L,W) ;
SPECS(SPEC,L,W).. SUM((KK,I), A(KK,SPEC)*X(KK,I,L,W)) =L= BW(L,SPEC) ;
PAYPRI(I,L,W).. SUM(KK, X(KK,I,L,W)) =L= 1 ;
ONEPRI(KK,L,W).. SUM(I, X(KK,I,L,W)) =L= 1 ;
CONSPRI(KK,I,L,W).. X(KK,I,L,W) =L= SUM(V, X(V,I-1,L,W)) + 1$(ORD(I) EQ 1) ;
CONSBUD(KK,I,L,W).. X(KK,I,L,W-1) =L= X(KK,I,L,W) ;
ONEREQ6(JS6,L,W).. SUM((KJ6,LW)$(ORD(LW) GT ORD(L) - %life% AND ORD(LW) LE ORD(L)),
         Y(KJ6,JS6,LW,W)) =L= 1 ;
ONEREQ7(JS7,L,W).. SUM((KJ7,LW)$(ORD(LW) GT ORD(L) - %life% AND ORD(LW) LE ORD(L)),
         Y(KJ7,JS7,LW,W)) =L= 1 ;
ONEREQ8(JS8,L,W).. SUM((KJ8,LW)$(ORD(LW) GT ORD(L) - %life% AND ORD(LW) LE ORD(L)),
         Y(KJ8,JS8,LW,W)) =L= 1 ;
ONEREQ9(JS9,L,W).. SUM((KJ9,LW)$(ORD(LW) GT ORD(L) - %life% AND ORD(LW) LE ORD(L)),
         Y(KJ9,JS9,LW,W)) =L= 1 ;
ONEREQ10(JS10,L,W).. SUM((KJ10,LW)$(ORD(LW) GT ORD(L) - %life% AND ORD(LW) LE ORD(L)),
         Y(KJ10,JS10,LW,W)) =L= 1 ;
ONESOLE6(L,W).. SUM((KJ6,I), X(KJ6,I,L,W)) =L= 1 ;
ONESOLE7(L,W).. SUM((KJ7,I), X(KJ7,I,L,W)) =L= 1 ;
ONESOLE8(L,W).. SUM((KJ8,I), X(KJ8,I,L,W)) =L= 1 ;
ONESOLE9(L,W).. SUM((KJ9,I), X(KJ9,I,L,W)) =L= 1 ;
ONESOLE10(L,W).. SUM((KJ10,I), X(KJ10,I,L,W)) =L= 1 ;
DEFZ1(JM1,L,W).. Z(JM1,L,W) =E= SUM((KJ1,I), X(KJ1,I,L,W)) ;
DEFZ2(JM2,L,W).. Z(JM2,L,W) =E= SUM((KJ2,I), X(KJ2,I,L,W)) ;
DEFZ3(JM3,L,W).. Z(JM3,L,W) =E= SUM((KJ3,I), X(KJ3,I,L,W)) ;
DEFZ4(JM4,L,W).. Z(JM4,L,W) =E= SUM((KJ4,I), X(KJ4,I,L,W)) ;
DEFZ5(JM5,L,W).. Z(JM5,L,W) =E= SUM((KJ5,I), X(KJ5,I,L,W)) ;
LINKXY6(KJ6,JS6,L,W).. Y(KJ6,JS6,L,W) =E= SUM(I, X(KJ6,I,L,W)) ;
```

```
LINKXY7(KJ7,JS7,L,W).. Y(KJ7,JS7,L,W) =E= SUM(I, X(KJ7,I,L,W)) ;
LINKXY8(KJ8,JS8,L,W).. Y(KJ8,JS8,L,W) =E= SUM(I, X(KJ8,I,L,W)) ;
LINKXY9(KJ9,JS9,L,W).. Y(KJ9,JS9,L,W) =E= SUM(I, X(KJ9,I,L,W)) ;
LINKXY10(KJ10,JS10,L,W).. Y(KJ10,JS10,L,W) =E= SUM(I, X(KJ10,I,L,W)) ;
DEFIZ1(JM1,L,W).. SUM(P,IZ(JM1,P,L,W)) =E= SUM(LW$(ORD(LW) GT ORD(L) - %life%
         AND ORD(LW) LE ORD(L)), Z(JM1,LW,W))  ;
DEFIZ2(JM2,L,W).. SUM(P,IZ(JM2,P,L,W)) =E= SUM(LW$(ORD(LW) GT ORD(L) - %life%
         AND ORD(LW) LE ORD(L)), Z(JM2,LW,W))  ;
DEFIZ3(JM3,L,W).. SUM(P,IZ(JM3,P,L,W)) =E= SUM(LW$(ORD(LW) GT ORD(L) - %life%
         AND ORD(LW) LE ORD(L)), Z(JM3,LW,W))  ;
DEFIZ4(JM4,L,W).. SUM(P,IZ(JM4,P,L,W)) =E= SUM(LW$(ORD(LW) GT ORD(L) - %life%
         AND ORD(LW) LE ORD(L)), Z(JM4,LW,W))  ;
DEFIZ5(JM5,L,W).. SUM(P,IZ(JM5,P,L,W)) =E= SUM(LW$(ORD(LW) GT ORD(L) - %life%
         AND ORD(LW) LE ORD(L)), Z(JM5,LW,W))  ;
DEFIY6(JS6,L,W).. IY(JS6,L,W) =L= SUM((KJ6,LW)$(ORD(LW) GT ORD(L) - %life%
         AND ORD(LW) LE ORD(L)), Y(KJ6,JS6,LW,W))  ;
DEFIY7(JS7,L,W).. IY(JS7,L,W) =L= SUM((KJ7,LW)$(ORD(LW) GT ORD(L) - %life%
         AND ORD(LW) LE ORD(L)), Y(KJ7,JS7,LW,W))  ;
DEFIY8(JS8,L,W).. IY(JS8,L,W) =L= SUM((KJ8,LW)$(ORD(LW) GT ORD(L) - %life%
         AND ORD(LW) LE ORD(L)), Y(KJ8,JS8,LW,W))  ;
DEFIY9(JS9,L,W).. IY(JS9,L,W) =L= SUM((KJ9,LW)$(ORD(LW) GT ORD(L) - %life%
         AND ORD(LW) LE ORD(L)), Y(KJ9,JS9,LW,W))  ;
DEFIY10(JS10,L,W).. IY(JS10,L,W) =L= SUM((KJ10,LW)$(ORD(LW) GT ORD(L) - %life%
         AND ORD(LW) LE ORD(L)), Y(KJ10,JS10,LW,W))  ;
XNK(NK,I,L,W).. X(NK,I,L,W) =E= 0;

option limrow=6;
MODEL SATELLITE /ALL/ ;

SATELLITE.OPTCR = 0.01;

$if exist matdata.gms $include matdata.gms

SOLVE SATELLITE USING MIP MAXIMIZING T ;


SCALAR COMPTIME time in CPU seconds to solve the model using CPLEX ;
COMPTIME = SATELLITE.RESUSD;

$libinclude matout X.l K I L W
$libinclude matout T.l
$libinclude matout COMPTIME

```

```
305 DISPLAY X.L;
306 DISPLAY COMPTIME;

  1 ************************************************************************
  2 *
  3 * AUTHOR: Capt Ben Kallemyn
  4 *         AFIT/ENS
  5 *         March 2007
  6 *
  7 * This program solves the multiple-launch memoryless prioritization problem
  8 *
  9 * The following assumptions/limitations hold for this program:
 10 *        1. There is a maximum of 10 requirements evenly split between
 11 *           sole-source and multi-source types.
 12 *        2. Associated with each requirement is space for 10 payloads.
 13 *           If there is a need for fewer, fill in the spaces with zeros.
 14 *        3. The reward functions for the multi-source requirements are
 15 *           concave, piecewise linear. The slopes must be non-increasing.
 16 *        4. If a payload is available, it is available for all launches.
 17 *           There should be a cost for each launch for each included payload.
 18 *
 19 ************************************************************************
 20
 21 *This sets the solver to CPLEX
 22 OPTIONS MIP=CPLEX;
 23
 24 *This sets the global variable for the # of payloads
 25 $setglobal pay 6
 26 $setglobal launch 2
 27 $setglobal life 2
 28 $setglobal pw 4
 29 *pw is the number of reqs that can be in const (# payloads per req * life)
 30 $if exist matglobs.gms $include matglobs.gms
 31
 32 *Define the sets
 33 SETS
 34 K available payload spaces / K1*K100 /
 35 I priority levels / 1 * %pay% /
 36 J requirements / J1*J10 /
 37 L launches / 1 * %launch% /
 38 W budget scenarios / W1*W3 /
 39 SPEC engineering specifications / WEI, POW, VOL /
 40 KK(K) payloads which are actually available
 41 V(K) another name for the set of payloads
```

```
JM1(J) multi-source requirement 1
JM2(J) multi-source requirement 2
JM3(J) multi-source requirement 3
JM4(J) multi-source requirement 4
JM5(J) multi-source requirement 5
JS(J) sole-source requirements / J6*J10 /
JS6(JS) sole-source requirement 1
JS7(JS) sole-source requirement 2
JS8(JS) sole-source requirement 3
JS9(JS) sole-source requirement 4
JS10(JS) sole-source requirement 5
KJ(K) the set of payloads which satisfy the sole-source reqs
KJ1(K) the set of payloads which satisfy req 1
KJ2(K) the set of payloads which satisfy req 2
KJ3(K) the set of payloads which satisfy req 3
KJ4(K) the set of payloads which satisfy req 4
KJ5(K) the set of payloads which satisfy req 5
KJ6(K) the set of payloads which satisfy req 6
KJ7(K) the set of payloads which satisfy req 7
KJ8(K) the set of payloads which satisfy req 8
KJ9(K) the set of payloads which satisfy req 9
KJ10(K) the set of payloads which satisfy req 10
P counter for peicewise linear functions / 1 * %pw% /
NK(K);

ALIAS (L,LW) ;
V(K) = YES;

PARAMETERS
Q(W) chance of budget scenario w
       / W1  .2
         W2  .6
         W3  .2  / ;

TABLE C(K,L)
                    1        2
         K1         400      400
         K2         200      300
         K11        600      700
         K12        500      600
         K51        800      800
         K52        950      900 ;

```

```
*Define set KK - the set of available payloads
LOOP(K$(C(K,'1') NE 0), KK(K) = YES; V(K) = YES);
LOOP(K$(C(K,'1') EQ 0), NK(K) = YES);

*Define the Kj subsets - the subset of payloads which satisfy each req
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE  1 AND ORD(K) LE  10), KJ1(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 11 AND ORD(K) LE  20), KJ2(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 21 AND ORD(K) LE  30), KJ3(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 31 AND ORD(K) LE  40), KJ4(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 41 AND ORD(K) LE  50), KJ5(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 51 AND ORD(K) LE  60), KJ6(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 61 AND ORD(K) LE  70), KJ7(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 71 AND ORD(K) LE  80), KJ8(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 81 AND ORD(K) LE  90), KJ9(K) = YES);
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 91 AND ORD(K) LE 100), KJ10(K)= YES);

LOOP(KJ1$(C(KJ1,'1') NE 0), JM1("J1") = YES);
LOOP(KJ2$(C(KJ2,'1') NE 0), JM2("J2") = YES);
LOOP(KJ3$(C(KJ3,'1') NE 0), JM3("J3") = YES);
LOOP(KJ4$(C(KJ4,'1') NE 0), JM4("J4") = YES);
LOOP(KJ5$(C(KJ5,'1') NE 0), JM5("J5") = YES);
LOOP(KJ6$(C(KJ6,'1') NE 0), JS6("J6") = YES);
LOOP(KJ7$(C(KJ7,'1') NE 0), JS7("J7") = YES);
LOOP(KJ8$(C(KJ8,'1') NE 0), JS8("J8") = YES);
LOOP(KJ9$(C(KJ9,'1') NE 0), JS9("J9") = YES);
LOOP(KJ10$(C(KJ10,'1') NE 0), JS10("J10") = YES);

*Define set KJ - the set of payloads satisfying sole-source reqs
LOOP(K$(C(K,'1') NE 0 AND ORD(K) GE 51), KJ(K)= YES);

TABLE A(K,SPEC) payload data
                    WEI      POW      VOL
         K1         200      350       3
         K2         100      450       2
         K11        400      300       4
         K12        300      500       4
         K51        900      750       9
         K52        800      700       7  ;

TABLE R(JS,L)
             1  2
         J6  14 21
         J7  0  0
```

```
128          J8  0  0
129          J9  0  0
130          J10 0  0  ;
131
132
133
134 TABLE BW(L,SPEC)
135              WEI  POW  VOL
136           1  1900 2600 22
137           2  1500 2300 18 ;
138
139 TABLE B(L,W)
140              W1   W2   W3
141           1  1500 2000 2500
142           2  1200 1600 2000 ;
143
144 TABLE BETA1(P,J,L)
145                  J1.1     J1.2
146          1        8        13
147          2        7        7
148          3        5        5
149          4        4        3 ;
150
151 TABLE BETA2(P,J,L)
152                  J2.1     J2.2
153          1        10       7
154          2        5        7
155          3        5        5
156          4        4        3 ;
157
158 TABLE BETA3(P,J,L)
159                  J3.1     J3.2
160          1        0        0
161          2        0        0 ;
162
163 TABLE BETA4(P,J,L)
164                  J4.1     J4.2
165          1        0        0
166          2        0        0  ;
167
168 TABLE BETA5(P,J,L)
169                  J5.1     J5.2
170          1        0        0
```

```
171          2        0        0  ;
172
173 VARIABLES
174 X(K,I,L,W) if payload k has priority level i and i is funded under scenario w
175 Y(K,J,L,W) if payload k satisfies requiremnt j under scenario w
176 IY(J,L,W) if sole-source req j is satisfied in const at launch l
177 Z(J,L,W) number of payloads which satisfy multi-source req j on launch l
178 IZ(J,P,L,W) number of payloads which satisfy multi-source req j in const at l
179 T total expected reward ;
180
181 BINARY VARIABLES X,Y,IY,IZ;
182
183 set ld(l);
184
185 EQUATIONS
186 REWARD define objective function
187 BUDGET(L,W) limit budget
188 SPECS(SPEC,L,W) limit specifications
189 PAYPRI(I,L,W) each priority level gets only 1 payload
190 ONEPRI(K,L,W) each payload has at most 1 priority level
191 CONSPRI(K,I,L,W) consecutive priority levels must be on list
192 CONSBUD(K,I,L,W) consecutive budget scenarios mmust be on list
193 ONEREQ6(JS,L,W)
194 ONEREQ7(JS,L,W)
195 ONEREQ8(JS,L,W)
196 ONEREQ9(JS,L,W)
197 ONEREQ10(JS,L,W)
198 ONESOLE6(L,W)
199 ONESOLE7(L,W)
200 ONESOLE8(L,W)
201 ONESOLE9(L,W)
202 ONESOLE10(L,W)
203 DEFZ1(J,L,W) define decision variable z for req 1
204 DEFZ2(J,L,W) define decision variable z for req 2
205 DEFZ3(J,L,W) define decision variable z for req 3
206 DEFZ4(J,L,W) define decision variable z for req 4
207 DEFZ5(J,L,W) define decision variable z for req 5
208 LINKXY6(K,J,L,W) link decision variable y to x for req 6
209 LINKXY7(K,J,L,W) link decision variable y to x for req 7
210 LINKXY8(K,J,L,W) link decision variable y to x for req 8
211 LINKXY9(K,J,L,W) link decision variable y to x for req 9
212 LINKXY10(K,J,L,W) link decision variable y to x for req 10
213 DEFIZ1(J,L,W)
```

```
214 DEFIZ2(J,L,W)
215 DEFIZ3(J,L,W)
216 DEFIZ4(J,L,W)
217 DEFIZ5(J,L,W)
218 DEFIY6(J,L,W)
219 DEFIY7(J,L,W)
220 DEFIY8(J,L,W)
221 DEFIY9(J,L,W)
222 DEFIY10(J,L,W)
223 XNK(K,I,L,W) ;
224
225
226 REWARD.. T =E= SUM(W, Q(W)*SUM(L$(LD(L)),(
227          SUM((JM1,P),BETA1(P,JM1,L)*IZ(JM1,P,L,W)) +
228          SUM((JM2,P),BETA2(P,JM2,L)*IZ(JM2,P,L,W)) +
229          SUM((JM3,P),BETA3(P,JM3,L)*IZ(JM3,P,L,W)) +
230          SUM((JM4,P),BETA4(P,JM4,L)*IZ(JM4,P,L,W)) +
231          SUM((JM5,P),BETA5(P,JM5,L)*IZ(JM5,P,L,W)) +
232          SUM(JS6,R(JS6,L)*IY(JS6,L,W)) +
233          SUM(JS7,R(JS7,L)*IY(JS7,L,W)) +
234          SUM(JS8,R(JS8,L)*IY(JS8,L,W)) +
235          SUM(JS9,R(JS9,L)*IY(JS9,L,W)) +
236          SUM(JS10,R(JS10,L)*IY(JS10,L,W))  ))) ;
237 BUDGET(L,W)$(LD(L)).. SUM((KK,I), C(KK,L)*X(KK,I,L,W)) =L= B(L,W) ;
238 SPECS(SPEC,L,W)$(LD(L)).. SUM((KK,I), A(KK,SPEC)*X(KK,I,L,W)) =L= BW(L,SPEC) ;
239 PAYPRI(I,L,W)$(LD(L)).. SUM(KK, X(KK,I,L,W)) =L= 1 ;
240 ONEPRI(KK,L,W)$(LD(L)).. SUM(I, X(KK,I,L,W)) =L= 1 ;
241 CONSPRI(KK,I,L,W)$(LD(L)).. X(KK,I,L,W) =L= SUM(V, X(V,I-1,L,W)) + 1$(ORD(I) EQ 1) ;
242 CONSBUD(KK,I,L,W)$(LD(L)).. X(KK,I,L,W-1) =L= X(KK,I,L,W) ;
243 ONEREQ6(JS6,L,W)$(LD(L)).. SUM((KJ6,LW)$(ORD(LW) GT ORD(L) - %life% AND ORD(LW) LE ORD(L)),
244          Y(KJ6,JS6,LW,W)) =L= 1 ;
245 ONEREQ7(JS7,L,W)$(LD(L)).. SUM((KJ7,LW)$(ORD(LW) GT ORD(L) - %life% AND ORD(LW) LE ORD(L)),
246          Y(KJ7,JS7,LW,W)) =L= 1 ;
247 ONEREQ8(JS8,L,W)$(LD(L)).. SUM((KJ8,LW)$(ORD(LW) GT ORD(L) - %life% AND ORD(LW) LE ORD(L)),
248          Y(KJ8,JS8,LW,W)) =L= 1 ;
249 ONEREQ9(JS9,L,W)$(LD(L)).. SUM((KJ9,LW)$(ORD(LW) GT ORD(L) - %life% AND ORD(LW) LE ORD(L)),
250          Y(KJ9,JS9,LW,W)) =L= 1 ;
251 ONEREQ10(JS10,L,W)$(LD(L)).. SUM((KJ10,LW)$(ORD(LW) GT ORD(L) - %life% AND ORD(LW) LE ORD(L)),
252          Y(KJ10,JS10,LW,W)) =L= 1 ;
253 ONESOLE6(L,W)$(LD(L)).. SUM((KJ6,I), X(KJ6,I,L,W)) =L= 1 ;
254 ONESOLE7(L,W)$(LD(L)).. SUM((KJ7,I), X(KJ7,I,L,W)) =L= 1 ;
255 ONESOLE8(L,W)$(LD(L)).. SUM((KJ8,I), X(KJ8,I,L,W)) =L= 1 ;
256 ONESOLE9(L,W)$(LD(L)).. SUM((KJ9,I), X(KJ9,I,L,W)) =L= 1 ;
```

```
257 ONESOLE10(L,W)$(LD(L)).. SUM((KJ10,I), X(KJ10,I,L,W)) =L= 1 ;
258 DEFZ1(JM1,L,W)$(LD(L)).. Z(JM1,L,W) =E= SUM((KJ1,I), X(KJ1,I,L,W)) ;
259 DEFZ2(JM2,L,W)$(LD(L)).. Z(JM2,L,W) =E= SUM((KJ2,I), X(KJ2,I,L,W)) ;
260 DEFZ3(JM3,L,W)$(LD(L)).. Z(JM3,L,W) =E= SUM((KJ3,I), X(KJ3,I,L,W)) ;
261 DEFZ4(JM4,L,W)$(LD(L)).. Z(JM4,L,W) =E= SUM((KJ4,I), X(KJ4,I,L,W)) ;
262 DEFZ5(JM5,L,W)$(LD(L)).. Z(JM5,L,W) =E= SUM((KJ5,I), X(KJ5,I,L,W)) ;
263 LINKXY6(KJ6,JS6,L,W)$(LD(L)).. Y(KJ6,JS6,L,W) =E= SUM(I, X(KJ6,I,L,W)) ;
264 LINKXY7(KJ7,JS7,L,W)$(LD(L)).. Y(KJ7,JS7,L,W) =E= SUM(I, X(KJ7,I,L,W)) ;
265 LINKXY8(KJ8,JS8,L,W)$(LD(L)).. Y(KJ8,JS8,L,W) =E= SUM(I, X(KJ8,I,L,W)) ;
266 LINKXY9(KJ9,JS9,L,W)$(LD(L)).. Y(KJ9,JS9,L,W) =E= SUM(I, X(KJ9,I,L,W)) ;
267 LINKXY10(KJ10,JS10,L,W)$(LD(L)).. Y(KJ10,JS10,L,W) =E= SUM(I, X(KJ10,I,L,W)) ;
268 DEFIZ1(JM1,L,W)$(LD(L)).. SUM(P,IZ(JM1,P,L,W)) =E= SUM(LW$(ORD(LW) GT ORD(L) - %life%
269          AND ORD(LW) LE ORD(L)), Z(JM1,LW,W))  ;
270 DEFIZ2(JM2,L,W)$(LD(L)).. SUM(P,IZ(JM2,P,L,W)) =E= SUM(LW$(ORD(LW) GT ORD(L) - %life%
271          AND ORD(LW) LE ORD(L)), Z(JM2,LW,W))  ;
272 DEFIZ3(JM3,L,W)$(LD(L)).. SUM(P,IZ(JM3,P,L,W)) =E= SUM(LW$(ORD(LW) GT ORD(L) - %life%
273          AND ORD(LW) LE ORD(L)), Z(JM3,LW,W))  ;
274 DEFIZ4(JM4,L,W)$(LD(L)).. SUM(P,IZ(JM4,P,L,W)) =E= SUM(LW$(ORD(LW) GT ORD(L) - %life%
275          AND ORD(LW) LE ORD(L)), Z(JM4,LW,W))  ;
276 DEFIZ5(JM5,L,W)$(LD(L)).. SUM(P,IZ(JM5,P,L,W)) =E= SUM(LW$(ORD(LW) GT ORD(L) - %life%
277          AND ORD(LW) LE ORD(L)), Z(JM5,LW,W))  ;
278 DEFIY6(JS6,L,W)$(LD(L)).. IY(JS6,L,W) =L= SUM((KJ6,LW)$(ORD(LW) GT ORD(L) - %life%
279          AND ORD(LW) LE ORD(L)), Y(KJ6,JS6,LW,W))  ;
280 DEFIY7(JS7,L,W)$(LD(L)).. IY(JS7,L,W) =L= SUM((KJ7,LW)$(ORD(LW) GT ORD(L) - %life%
281          AND ORD(LW) LE ORD(L)), Y(KJ7,JS7,LW,W))  ;
282 DEFIY8(JS8,L,W)$(LD(L)).. IY(JS8,L,W) =L= SUM((KJ8,LW)$(ORD(LW) GT ORD(L) - %life%
283          AND ORD(LW) LE ORD(L)), Y(KJ8,JS8,LW,W))  ;
284 DEFIY9(JS9,L,W)$(LD(L)).. IY(JS9,L,W) =L= SUM((KJ9,LW)$(ORD(LW) GT ORD(L) - %life%
285          AND ORD(LW) LE ORD(L)), Y(KJ9,JS9,LW,W))  ;
286 DEFIY10(JS10,L,W)$(LD(L)).. IY(JS10,L,W) =L= SUM((KJ10,LW)$(ORD(LW) GT ORD(L) - %life%
287          AND ORD(LW) LE ORD(L)), Y(KJ10,JS10,LW,W))  ;
288 XNK(NK,I,L,W)$(LD(L)).. X(NK,I,L,W) =E= 0;
289
290 option limrow=6;
291 MODEL SATELLITE /ALL/ ;
292
293 SATELLITE.OPTCR = 0.01;
294
295 $if exist matdata.gms $include matdata.gms
296
297 set iter /1* %launch% /;
298
299 loop(iter,
```

```
300
301 ld(l)=no;
302 ld(l)$(ord(l) le ord(iter))=yes;
303
304 SOLVE SATELLITE USING MIP MAXIMIZING T ;
305
306 x.fx(k,i,ld,w)=x.l(k,i,ld,w);
307
308 );
309
310 SCALAR COMPTIME time in CPU seconds to solve the model using CPLEX ;
311 COMPTIME = SATELLITE.RESUSD;
312
313 $libinclude matout X.l K I L W
314 $libinclude matout T.l
315 $libinclude matout COMPTIME
316
317 DISPLAY X.L;
318 DISPLAY COMPTIME;
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
23–03–2007	Master's Thesis	Mar 2006 — Mar 2007

4. TITLE AND SUBTITLE

Prioritizing Satellite Payload Selection via Optimization

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Kallemyn, Benjamin S., Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/07-14

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Reconnaissance Office
Attn: William J. Comstock
Welkin Associates, Ltd.
4801 Stonecroft Blvd., Suite 210
Chantilly, VA 20151; COMM: (703)863-8163

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT This thesis develops optimization models for prioritizing payloads for inclusion on satellite buses with volume, power, weight and budget constraints. The first model considers a single satellite launch for which the budget is uncertain and constellation requirements are not considered. Subsequently, we include constellation requirements and provide a more enhanced model. Both single-launch models provide a prioritized list of payloads to include on the launch before the budget is realized. The single-launch models are subsequently extended to a sequence of multiple launches in two cases, both of which incorporate an explicit dependence on the constellation composition at each launch epoch. The first case ignores future launches and solves a series of independent single-launch problems. The second case considers all launches simultaneously. The optimization models for single- and multiple-launch cases are evaluated through a computational study. It was found that, when the budget distribution is skewed, the prioritization model outperforms a greedy payload selection heuristic in the single-launch model. For the multiple-launch models, it was found that the consideration of future launches can significantly improve the objective function values.

15. SUBJECT TERMS

Prioritization, Satellite payloads, Optimization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jeffrey P. Kharoufeh, Phd, (ENS)
U	U	U	UU	128	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4603; jeffrey.kharoufeh@afit.edu